TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP – THÁI NGUYÊN

KHOA ĐIỆN TỬ

NGUYỄN TUẤN ANH

Email: **tuananhk43@yahoo.com**

ĐT: 0912662003

LẬP TRÌNH ỨNG DỤNG TRÊN POCKET PC

THÁI NGUYÊN 05/2006

MỤC LỤC

LẬP TRÌNH CHO THIẾT BỊ DI ĐỘNG TRÊN NỀN WINDOWS MOBILE

Sau đây chúng ta sẽ tìm hiểu lập trình cho thiết bị di động trên nền Windows mobile. Trong tài liệu này các ví dụ được triển khai bằng ngôn ngữ lập trình C#, trong **Visual Studio .NET 2003**.

Chương 1 Thiết lập môi trường phát triển ứng dụng

1.1 Smart Device Extensions và .NET Compact Framework

1.1.1 Yêu cầu hệ thống

Smart Device Extensions là môi trường phát triển tích hợp (IDE) mà các nhà phát triển nhằm vào .NET Compact Framework. Nó là một thành phần củaVisual Studio .NET version 7.1 hoặc mới hơn.

Để chạy được các công cụ trên, yêu cầu tối thiểu về cấu hình như sau:

Bảng 1.1. yêu cầu hệ thống cho Visual Studio .NET 2003

Lĩnh vực	**Yêu cầu**
Operating system and RAM	Windows 2000 Professional; 96MB RAM, 128MB đề nghị Windows 2000 Server; 192MB RAM, 256MB đề nghị Windows XP Professional; 192MB RAM, 256MB đề nghị Windows XP Home; 96MB RAM, 128MB đề nghị Windows .NET Server 2003; 192MB RAM, 256MB đề nghị
Hard disk space	Ít nhất 900MB trên ổ chứa hệ điều hành và khoảng 4.1GB để cài Micorsoft Visual Studio .Net
Processor speed	Tối thiểu Pentium II 450MHz hoặc tương đương; Pentium III 600MHz hoặc lớn hơn
Device connectivity	ActiveSync 3.5 hoặc mới hơn

Bạn cần phải có thiết bị để chạy thử chương trình. .NET Compact Framework tương thích với tất cả các thiết bị có khả năng chạy hệ điều hành Pocket PC.

1.1.2 Sử dụng Smart Device Extensions trong quá trình phát triển

Cách dễ nhất để phát triển .NET Compact Framework là sử dụng Smart Device Extensions trong Visual Studio .NET 7.1. Nó đơn giản là mở rộng của Visual Studio 7.1, Smart Device Extensions đưa ra các kiểu tạo ứng dụng, cho phép chúng ta tập chung vào các thiết bị sử dụng Windows CE hỗ trợ .NET Compact Framework, như là Pocket PC. Điều này có nghĩa là sử

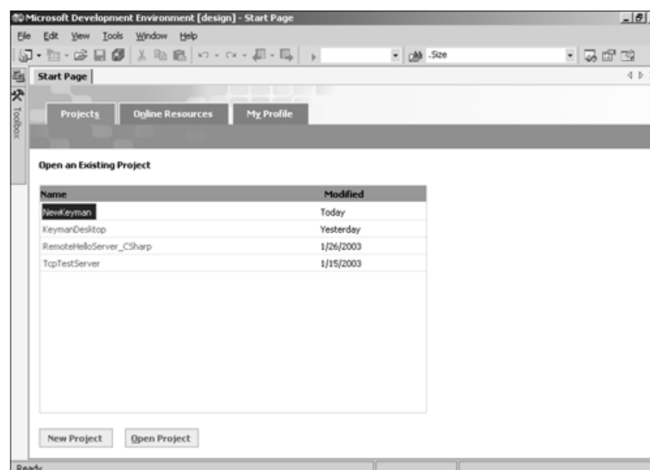
dụng Smart Device Extensions để phát triển các ứng dụng trên Windows CE như phát triển các ứng dụng trên Windows 2000 or XP.

Tạo ứng dụng cho các thiết bị Pocket PC

Chúng ta sẽ tạo một ứng dụng đơn giản "Hello World" bằng ngôn ngữ C#.

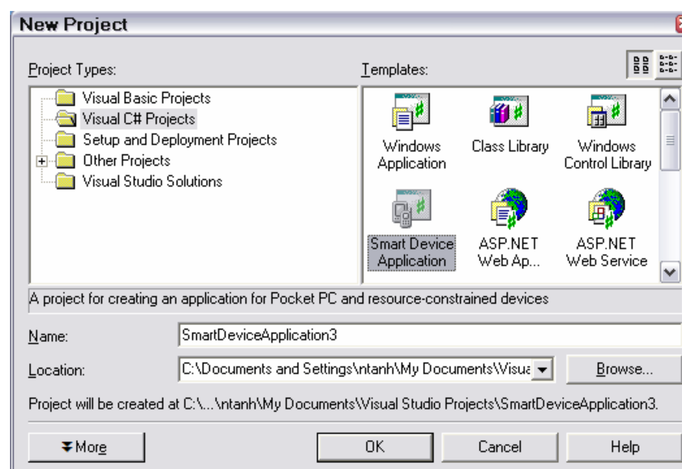
Bước 1: Khi chúng ta chạy Visual Studio .NET lần đâu, sẽ hiển thị Start Page, như hình 2. Để tạo ứng dụng mới, bấm vào nút có nhãn New Project gần phía dưới của màn hình. Hoặc vào menu File -> New -> Project hoặc sử dụng Ctrl+ Shift +N.

Hình 1.1. Start Page được hiển thị khi chạy Visual Studio .NET.



Bước 2: Sau khi chọn New Project, một hộp thoại xuất hiện ra cho phép chúng ta chọn kiểu dự án. Lựa chọn mục Visual C# Project và Smart Device Application như hình 1.2. Điền tên dự án vào mục Name, và nơi chứa dự án vào mục Localtion.

Hình 1.2. Hộp thoại tạo một Visual C# Smart Device Application



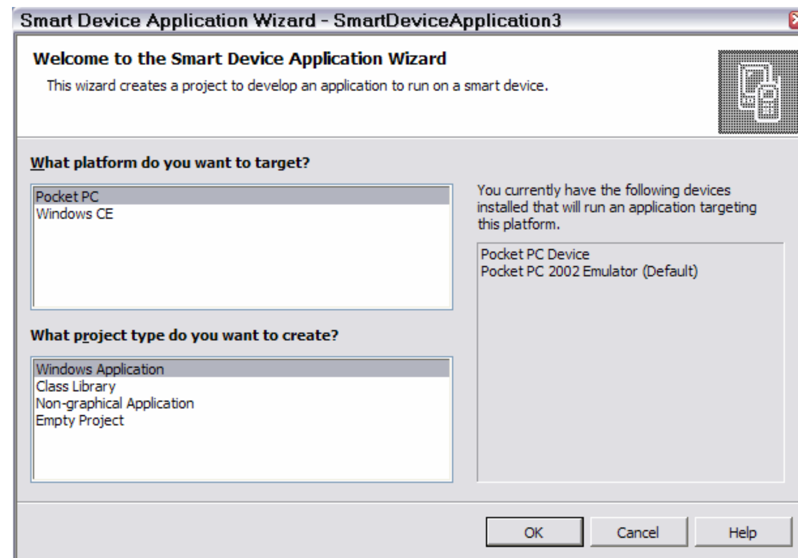
Bước 3: Hộp thoại tiếp theo như hình 1.3. Hộp thoại này chia làm hai phần:

- "What platform do you want to target?" Phần này cho phép chúng ta chọn kiểu thiết bị mà chúng ta muốn phát triển trên nó. Chúng ta sẽ chọn nền tảng Pocket PC, điều này có nghĩa

ứng dụng của chúng ta sẽ chạy trên tất cả các thiết bị hỗ trợ hệ điều hành Pocket PC, bao gồm cả SmartPhones.

- "What project type do you want to create?": Windows Application, Class Library, Non-graphical Application, và Empty Project. Chúng ta sẽ chọn Windows Application. Kiểu dự án này thiết lập form chính tự động và cung cấp môi trường thiết kế đồ họa giúp dễ dàng thêm các điều khiển vào ứng dụng.

Hình 1.3. Lựa chọn nền tảng và mẫu ứng dụng



- Class Library: sử dụng để tạo ra các thư viện liên kết động (DLL) cho .NET Compact Framework.

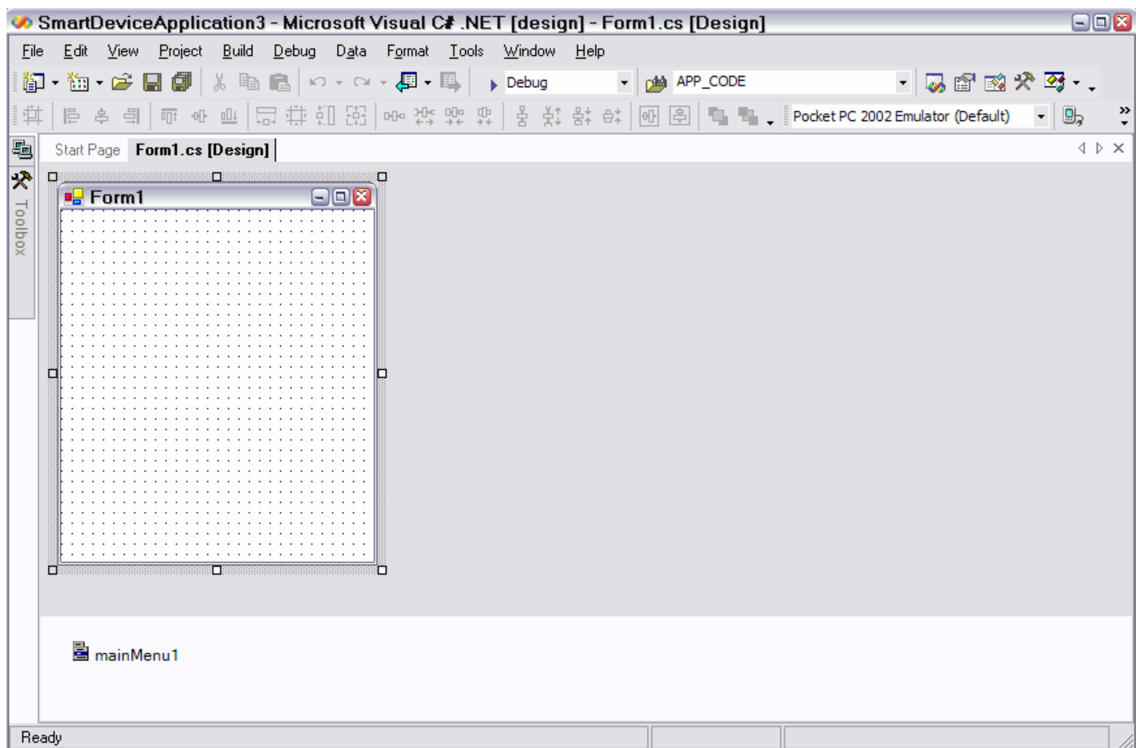
- Non-graphical Application: cho phép người sử dụng tạo ra các ứng dụng kiểu console, những ứng dụng loại này hữu dụng trên các thiết bị chạy hệ điều hành Windows CE cung cấp nhắc nhở dòng lệnh. Non-graphical Application thiết lập số lượng nhỏ nhất mã nguồn bắt đầu vì vậy người sử dụng có thể.

- Non-graphical Application: Tạo ứng dụng không dùng đồ họa.

- Empty Project: tạo một file mã nguồn rỗng. Khi đó người sử dụng phải tự đưa vào tất cả mã nguồn để thiết lập giao diện.

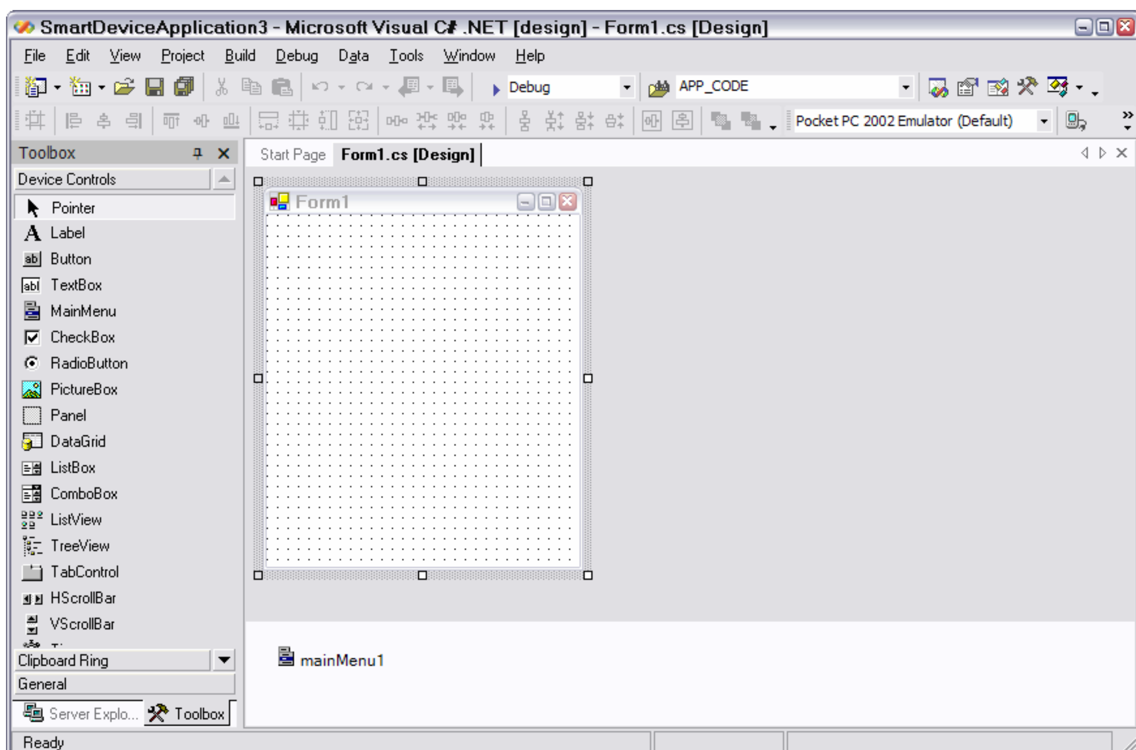
Bước 4: Sau khi bạn lựa chọn như hình 1.3, bấm OK. Visual Studio tự động kích hoạt Smart Device Extensions và đưa đến phần thiết kế Forms, như hình 1.4. Thiết kế Forms giống như thiết kế được sử dụng trong các dự án desktop.

Hình 1.4. Thiết kế Forms xuất hiện sau khi dự án được tạo



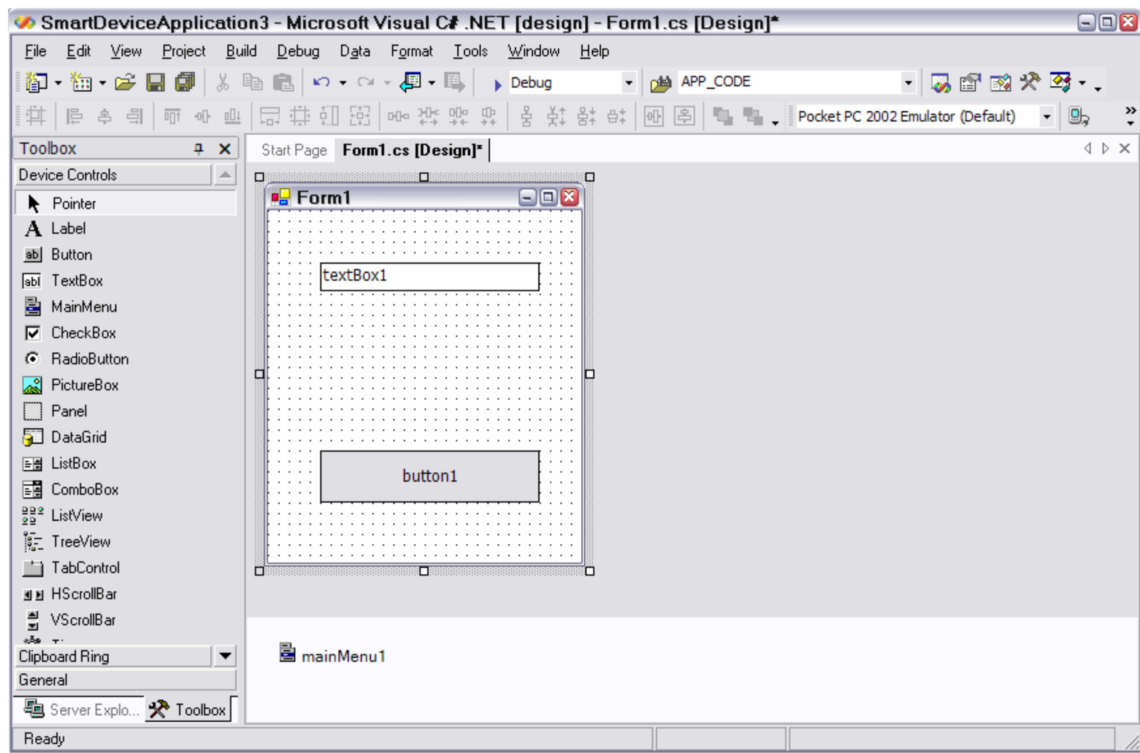
Bước 5: Bên trái của phần thiết kế Forms, là nhãn Toolbox. Bấm vào đó đưa đến cho chúng ta hộp công cụ Toolbox, như hình 1.5.

Hình 1.5. Hộp công cụ Toolbox cho dự án Smart Device Application



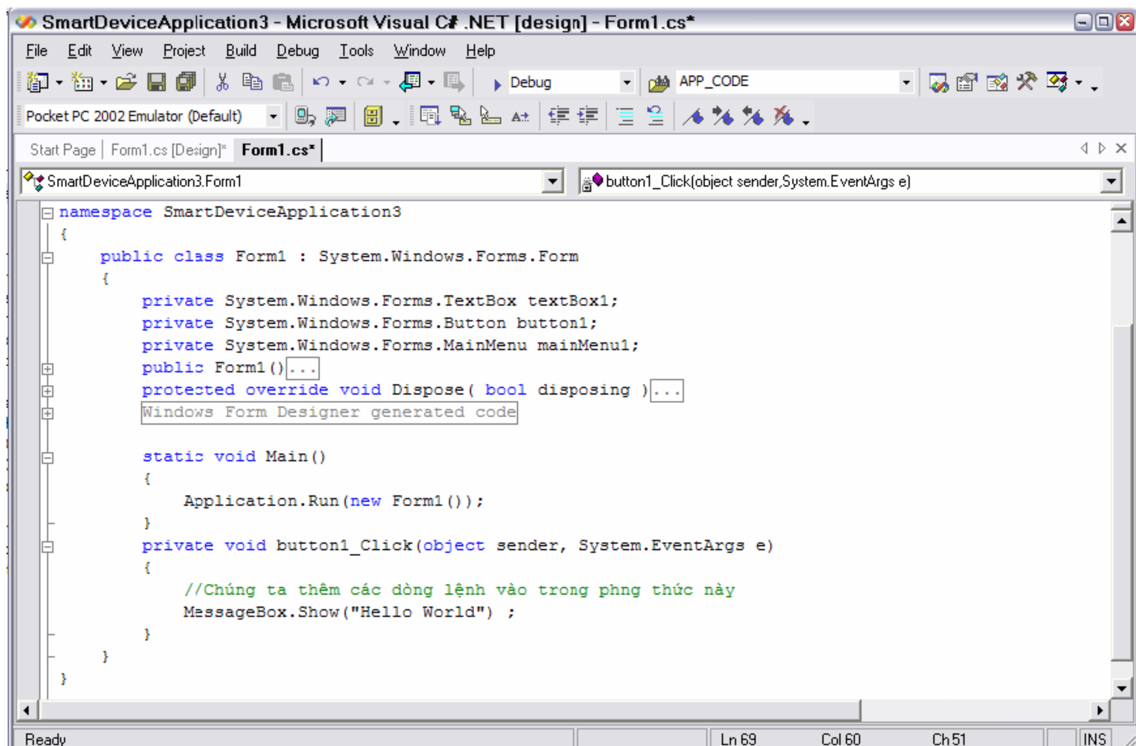
Bước 6: Tất cả các điều khiển trong hộp công cụ đều có thể sử dụng trong các dự án .NET Compact Framework. Kéo một số điều khiển vào Forms như hình 1.6.

Hình 1.7. Sau khi kéo một số điều khiển vào Forms.



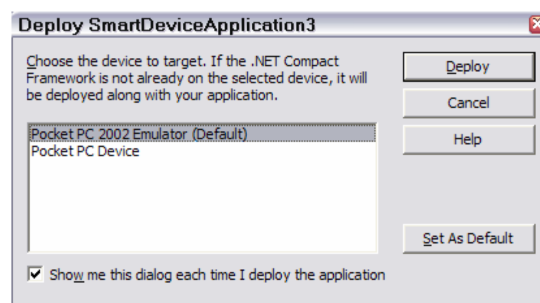
Bước 7: Bấm đúp vào nút có nhãn button1, IDE đưa đến phần soạn thảo mã nguồn và con trỏ sẽ nhấp nháy ở trong phương thức button1_Click. Chúng ta sẽ đưa vào một số dòng lệnh như hình 1.7.

Hình 1.7. Visual Studio hiển thị mã lệnh khi nút trong Form được bấm đúp.



Bước 8: Bây giờ chúng ta có thể biên dịch và triển khai trên thiết bị. Để triển khai trên thiết bị và chạy ứng dụng, chọn Debug, Start Without Debugging. Trước tiên Visual Studio biên dịch mã nguồn và đưa cho chúng ta hộp thoại Deploy SmartDeviceApplication, như hình 1.8.

Hình 1.8. Trước khi triển khai ứng dụng trên thiết bị, Visual Studio đưa ra hộp thoại.



Bước 9: Để chạy thử ứng dụng trên máy tính Desktop, chúng ta chọn Pocket PC 2002 Emulator. Nếu muốn chạy thử trên thiết bị thực, chúng ta chọn Pocket PC Device. Phải đảm bảo rằng thiết bị đã được kết nối thông quan đường ActiveSync trước khi triển khai ứng dụng trên thiết bị. Sau đó chọn Deploy.

Bước 10: Visual Studio cài đặt .NET Compact Framework và chạy ứng dụng. Nếu chúng ta bấm vào nút Button1, chúng ta nhận được thông báo “Hello World” như hình 1.9.

Hình 1.10. Thiết bị emulator chạy ứng dụng hello world.



1.2 Các thiết bị phi chuẩn

.NET Compact Framework có thể chạy trên rất nhiều các thiết bị phần cứng chạy Windows CE. Bảng 1.2 cho chúng ta thấy các bộ xử lý được hỗ trợ bởi .NET Compact Framework và các hệ điều hành hỗ trợ cho các bộ xử lý.

.NET Compact Framework được lưu trữ như là một file CAB trên máy Desktop. Chỉ có một file CAB duy nhất cho mỗi hệ điều hành và kiểu bộ xử lý mà .NET Compact Framework hỗ trợ. Smart Device Extensions đưa file CAB phù hợp vào thiết bị khi nó xác định thiết bị không cài đặt .NET Compact Framework. Trong phần này, chúng ta thảo luận chi tiết bộ xử lý làm việc như thế nào và làm thế nào để tự triển khai các file CAB nếu không thể triển khai tự động.

Tất cả các thiết bị Pocket PC chạy hệ điều hành Pocket PC version 2003 hoặc mới hơn đều có .NET Compact Framework trong ROM. Nếu chúng ta không thể triển khai hoặc gỡ lỗi ứng dụng trên các thiết bị, trong phần này chúng ta sẽ học cách làm thế nào để Smart Device Extensions kết nối với các thiết bị để gỡ lỗi và triển khai và thảo luận một vài vấn đề liên quan.

Bảng 1.2. Các bộ xử lý và hệ điều hành được .NET Compact Framework hỗ trợ

Tên CPU	**Phiên bản hệ điều hành hỗ trợ**
Intel ARM 4	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn
Intel ARM 4i	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn
Hitachi SH3	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn
Hitachi SH4	Pocket PC 2003 và WinCE 4.1 hoặc mới hơn

Intel 80x86	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn
MIPS 16	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn
MIPS II	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn
MIPS IV	Pocket PC 2000, 2002, 2003, và WinCE 4.1 hoặc mới hơn

Bảng 1.2 mô tả .NET Compact Framework chạy trên nhiều phần cứng. Có ba mức hỗ trợ cho các thiết bị phi chuẩn:

- Hỗ trợ đầy đủ triển khai và gỡ lỗi: mức hỗ trợ này có nghĩa IDE có thể triển khai cùng với thiết bị và gỡ lỗi mã nguồn đang chạy trên thiết bị.

- Hỗ trợ triển khai: có nghĩa IDE chỉ có thể triển khai trên thiết bị nhưng không thể gỡ lỗi chạy trên thiết bị.

- Hỗ trợ Target: có nghĩa là chúng ta có thể phát triển ứng dụng bằng Visual Studio nhưng chúng ta phải tự cài đặt Compact Framework trên thiết bị và sao chép vào thiết bị.

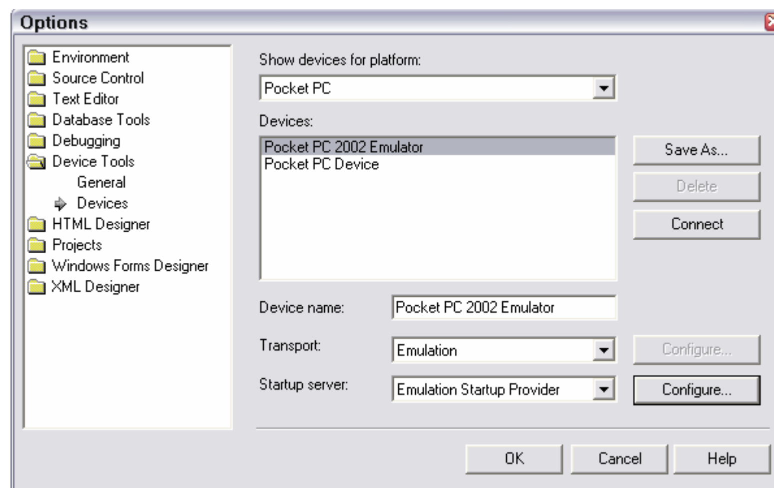
Kết nối Visual Studio với các thiết bị

Để thiết lập giao tiếp Visual Studio với thiết bị, chúng ta làm theo các bước sau:

Bước 1: Chọn Tools, Options trong Visual Studio.

Bước 2: Bấm đúp trên mục Device Tools và chọn Devices. Xem hình hình 1.11.

Hình 1.11. Sử dụng hộp thoại kết nối thiết bị để chọn kiểu thiết bị muốn kết nối.



Bước 3: Chọn nền tảng Pocket PC hay Windows CE.

Bước 4: Chọn kiểu thiết bị mà chúng ta muốn triển khai ứng dụng trên đó. Hình 1.11 cho phép chọn Emulator hoặc thiết bị Pocket PC.

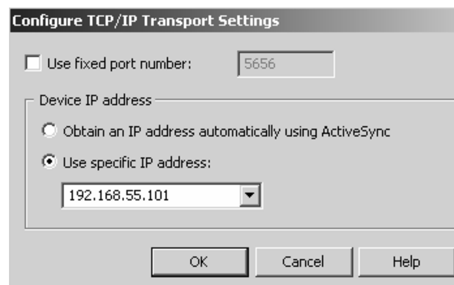
Bước 5: Lựa chọn cách thức truyền tin được dùng. Thiết bị Pocket PC có hai lựa chọn là: kết nối TCP và IRDA.

Kết nối TCP Connect Transport có nghĩa là thiết bị desktop sẽ kết nối với ConmanClient.exe trên thiết bị bằng kết nối TCP.

Kết nối IRDA Tranport sử dụng IRDA trên thiết bị để kết nối. Điều này rất hữu ích khi máy tính của chúng ta là laptop có cổng IRDA.

Bước 6: Nếu chọn TCP Connect Transport, sau đó bạn có thể thay đổi bằng cách chọn nút Configure… sau đó sẽ nhận được như hình 1.12.

Hình 1.12. TCP Connect Transport cho phép chúng ta thiết lập kết nối tới thiết bị TCP.



Bước 7: Hộp thoại như hình 12 cho phép chúng ta thiết lập địa chỉ IP cho thiết bị. Nếu thiết bị kết nối bằng ActiveSync, Visual Studio có thể tự động điền nhận ra địa chỉ. Bạn có thể lựa chọn sử dụng số hiệu cổng khác với cổng 5656. Để không sử dụng cổng mặc định, bạn phải tự cấu hình ConmanClient.exe trên thiết bị

Chương 2 Thiết kế các ứng dụng GUI bằng Windows Forms

2.1 Những điều khiển không hỗ trợ

Sau đây là danh sách các điều khiển không được .NET Compact Framework hỗ trợ.

- `CheckedListBox`
- `ColorDialog`
- `ErrorProvider`
- `FontDialog`
- `GroupBox`
- `HelpProvider`
- `LinkLabel`
- `NotificationBubble`
- `NotifyIcon`
- `All Print controls`
- `RichTextBox`
- `Splitter`

2.2 Những hàm .NET Compact Framework không hỗ trợ

Danh sách các hàm .NET Compact Framework không hỗ trợ.

- `AcceptButton`
- `CancelButton`
- `AutoScroll`
- `Anchor`
- Giao diện đa tài liệu (`MDI`)
- `KeyPreview`
- `TabIndex`
- `TabStop`
- Kéo thả
- Tất cả các khả năng in ấn
- Các điều khiển `Hosting ActiveX`

2.3 Thiết kế Form trên Visual Studio .NET

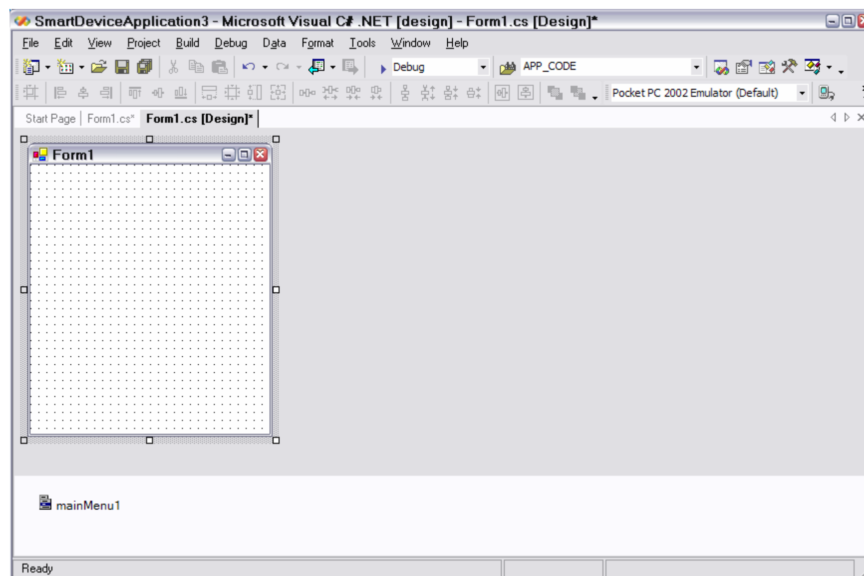
Thiết kế Form bằng Visual Studio .NET cho phép chúng ta thiết kế giao diện ứng dụng trực quan bằng cách kéo thả các điều khiển. Bạn có thể điều chỉnh vị trí các điều khiển, thiết lập các thuộc tính thông qua cửa sổ thuộc tính, và tạo các sự kiện cho các điều khiển.

2.3.1 Cửa sổ thiết kế Forms

Khi chúng ta tạo một dự án Smart Device Extension (SDE), là một ứng dụng cửa sổ, Visual Studio .NET sẽ mở dự án trong phần hiển thị thiết kế. Chúng ta có thể lựa chọn thiết kế từ menu View để đưa vào khung nhìn của dự án. Hình 2.1 đưa đến cho chúng ta Form Designer của dự án SDE Pocket PC trong khung nhìn Designer.

Chú ý rằng thành phần `mainMenu1` ở phía dưới của cửa sổ thiết kế. Khu thiết kế danh riêng cho các điều khiển, những điều khiển không có sự thể hiện trực quan, giống như là điều khiển `MainMenu`, điều khiển `ContextMenu`, điều khiển `Timer`, và còn nhiều điều khiển khác.

Hình 2.1. SDE Pocket PC trong màn hình Designer view

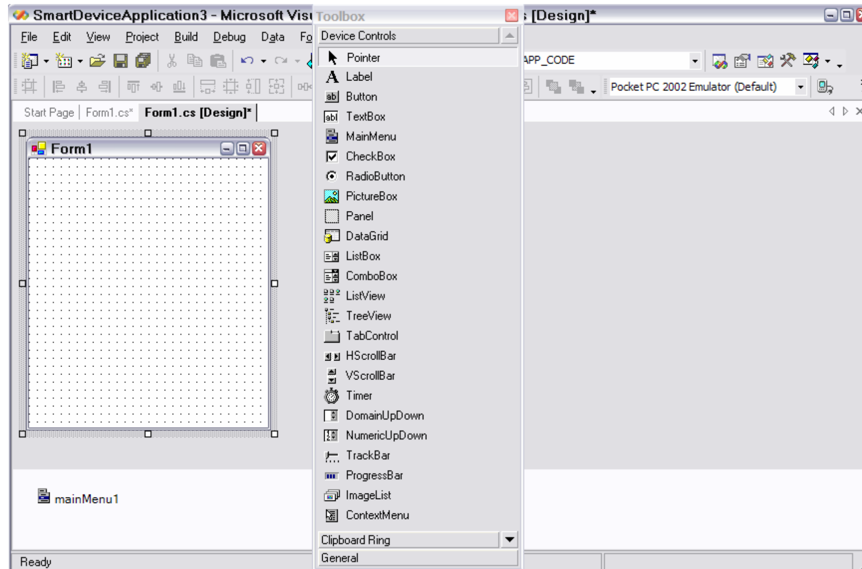


Khi Form Designer được sử dụng để xây dựng ứng dụng, phương thức `InitializeComponent` chứa đựng mã nguồn để xây dựng giao diện của ứng dụng. Mã nguồn này có ảnh hướng lớn đến quá trình thực hiện nếu form của bạn chứa đựng một vài điều khiển ẩn. Trên .NET Compact Framework đề nghị các cửa sổ được tạo theo hướng từ trên xuống. Ví dụ, nếu một panel được đặt trên form và panel đó chứa một vài điều khiển, panel đó sẽ được thêm vào form, và sau đó các điều khiển mới được thêm vào panel.

2.3.2 Cửa sổ ToolBox

Cửa sổ `ToolBox` chứa đựng tất cả các điều khiển của .NET Compact Framework mà chúng ta có thể thêm vào ứng dụng. Để thêm một điều khiển vào ứng dụng vào lúc thiết kế rất dễ như là kéo một điều khiển từ `ToolBox` và thả vào Forms của ứng dụng trong cửa sổ Form Designer. Hình 2.2

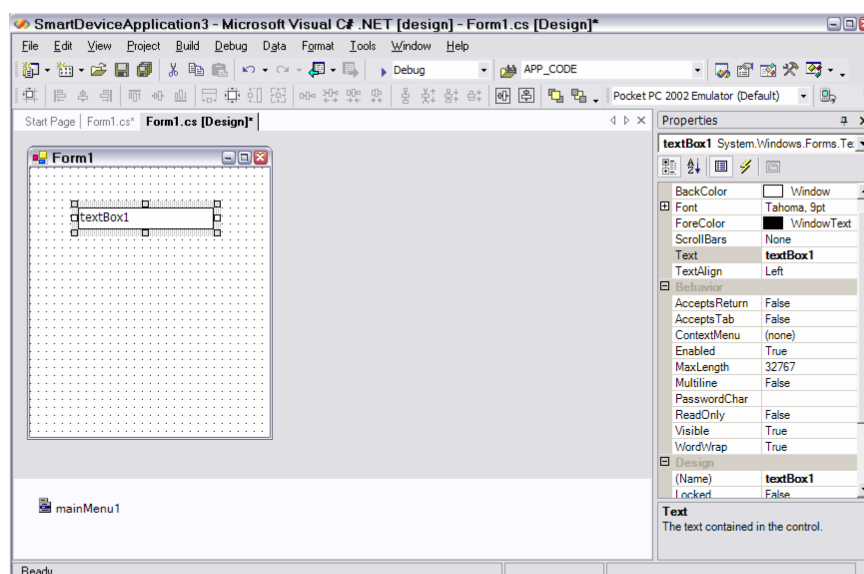
Hình 2.2. Cửa sổ ToolBox cho dự án SDE Pocket PC.



2.3.3 Cửa sổ thuộc tính

Cửa sổ thuộc tính chứa đựng tất cả các thuộc tính public của điều khiển đang lựa chọn trong cửa sổ Form Designer. Bạn có thể thay đổi thuộc tính của các điều khiển bằng cách đưa giá trị vào điều khiển `TextBox` bên cạnh các tên thuộc tính. Nếu thuộc tính có giới hạn số lượng giá trị, sau đó hộp thả xuốngđược hiển thị bên cạnh tên thuộc tính đó. Nếu giá trị của thuộc tính là một tập hợp các đối tượng hoặc một đối tượng phức tạp, có thể đặc tính đó ở bên cạnh tên thuộc tính. Chọn vào đặc tính đó sẽ hiển thị một hộp thoại cho phép chúng ta sửa đổi giá giá trị của thuộc tính. Hình 2.3 hiển thị cửa sổ thuộc tính khi một điều khiển `TextBox` được chọn.

Hình 2.3. Cửa sổ Properties của một điều khiển `TextBox`



2.4 Tìm hiểu các nền tảng Window Form

Các dự án Smart Device Extensions (SDE) phải nhằm vào hệ điều hành Pocket PC hoặc Windows CE .NET. Hai nền tảng có các hàm giao diện người sử dụng API khác nhau. Một dự án SDE thao tác bằng cách gọi các thư viện khác nhau cho mỗi nền tảng.

2.4.1 Nền tảng Windows CE .NET

Dự án Windows CE .NET giống như các dự án ứng dụng Window .NET Framework đầy đủ. Trước tiên, nút minimize, nút maximize, và nút close xuất hiện trong hộp điều khiển của ứng dụng như chúng ta làm việc trên đối tượng Form .NET Framework đầy đủ. Các nút này có hành vi như là trên desktop. Chúng ta có thể loại bỏ các nút đó bằng cách gán thuộc tính `ControlBox` của Form là false. Chúng ta cũng có thể loại bỏ nút minimize và nút maximize bằng cách thiết lập các thuộc tính `MinimizeBox` và `MaximizeBox` thành false.

Khi một form ứng dụng Windows CE .NET được tạo bằng phần thiết kế Form của Visual Studio.NET, kích cỡ được thiết lập là 640 x 450. Bạn có thể thay đổi thuộc tính Size nếu nó không phù hợp. Mặc dù lớp Form được đưa ra thuộc tính `FormBorderSytle`, thiết lập thuộc tính `Sizable` sẽ không ảnh hưởng tới đường viền của cửa sổ. Những ứng dụng Windows CE .NET không thể thay đổi kích cỡ. Nó chỉ có thể thu nhỏ, phóng to hết màn hình, hoặc kích cỡ như thuộc tính Size.

2.4.2 Nền tảng Pocket PC

Các ứng dụng Pocket PC trong tương lai sẽ theo hướng các dự án ứng dụng Windows .NET Framework đầy đủ. Trước tiên, một đối tượng `MainMenu` luôn luôn được thêm vào một ứng dụng Pocket PC. Chúng ta có thể loại bỏ menu đó, những hành động đó sẽ là nguyên nhân phát sinh ngoại lệ khi tương tác với Soft Input Panel (SIP). SIP là một phần mềm bổ sung của bàn phím QWERTY.

Cửa sổ `ToolBox` của Visual Studio .NET chứa đựng một điều khiển `InputPanel`. Trên mỗi Pocket PC điều khiển này cho phép chúng ta tương tác với SIP. `InputPanel` cho phép chúng ta nâng nên và hạ xuống SIP. `InputPanel` sẽ gắn vào ứng dụng khi SIP có khả năng. Trong Form phải có một điều khiển `MainMenu` hợp lệ cho điều khiển `InputPanel` được thêm vào trong Form. Nếu không có điều khiển `MainMenu` trên Form, sau đó một ngoại lệ sẽ được đưa ra vào lúc thực thi khi chúng ta cố gắn hiện `InputPanel`.

2.5 Làm việc với Form

Điều khiển Form là nơi chứa các điều khiển của ứng dụng. Điều khiển Form hiện diện là một cửa sổ chứa các điều khiển của ứng dụng. Lớp Form có nhiều thuộc tính tạo ra hành vi khác nhau phụ thuộc vào nền tảng (target platform).

2.5.1 Ảnh hưởng của thuộc tính FormBorderStyle

Thuộc tính `FormBorderSytle` xác định kiểu đường viền của Form. Giá trị mặc định là `FormBorderStyle.FixedSingle`.

Trên Pocket PC, thiết lập thuộc tính `FormBorderStyle.None` để tạo một form cùng với đường viên và không có tiêu đề. Kiểu Form này có thể thay đổi kích thước và di chuyển trong mã nguồn nhưng không thể thay đổi bởi người sử dụng. Thiết lập thuộc tính `FillBorderStyle.FixedSingle` hoặc bất kỳ giá trị nào khác sẽ tạo ra một Form bao trùm toàn bộ màn hình, và Form sẽ không thể di chuyển và thay đổi kích thước.

Trên Windows CE .NET, thiết lập thuộc tính `FormBorderStyle.FixedDialog` hoặc `FormBorderStyle.None` sẽ tạo ra một form không có đường viền và tiêu đề. Form sẽ di chuyển và thay đổi kích thước chỉ thông qua mã nguồn của chương trình. Thiết lập thuộc tính `FormBorderStyle.FixedSingle` hoặc bất kỳ giá trị nào khác sẽ tạo Form có một kích cỡ trả về thông qua thuộc tính Size với đường viên và tiêu đề. Form chỉ có thể thay đổi kích thước và di chuyển thông qua mã nguồn, và người sử dụng sẽ có thể di chuyển form.

2.5.2 Sử dụng thuộc tính ControlBox

Thuộc tính `ControlBox` của Form xác định hộp điều khiển của Forms có được hiển thị hay . Thiết lập thuộc tính `ControlBox` thành `true` sẽ hiển thị hộp điều khiển. Thiết lập thuộc tính này thành `false` sẽ ẩn hộp điều khiển.

2.5.3 Thuộc tính MinimizeBox và MaximizeBox

Trên Pocket PC hộp điều khiển chỉ chứa đựng nhiều nhất một nút, một là nút minimize, nhãn X, hoặc nút close, nhãn OK. Trên Windows CE .NET hộp điều khiển có thể chứa đựng nút minimize, nút maximize, và nút close. Để các nút này hiển thị được điều khiển bằng thuộc tính `MinimizeBox` và `MaximizeBox`. Bảng 2.1 mô tả giá trị vị trí của `MinimizeBox` và ảnh hưởng của mỗi nền tảng. Bảng 2.3.

Bảng 2.1. Giá trị thuộc tính `MinimizeBox` và ảnh hưởng của nó cho mỗi nền tảng

Giá trị	Ứng dụng POCKET PC	Ứng dụng WINDOWS CE .NET
`True`	**X** (nút minimize trên menu bar)	Nút minimize giống như thông thường
`False`	**OK** (nút close trên menu bar)	Không có nút minimize trên thanh tiêu đề

Bảng 2.2. Giá trị thuộc tính `MaximizeBox` và ảnh hưởng của nó cho mỗi nền tảng

Giá trị	Ứng dụng POCKET PC	Ứng dụng WINDOWS CE .NET
`Normal`	Ứng dụng sẽ điền đầy vùng desktop, cái mà toàn bộ vùng màn hình trừ phần menu start và vùng thanh menu chính.	Ứng dụng có kích cỡ như thuộc tính Size
`Maximize`	Ứng dụng điền đầu màn hình. Nó sẽ ẩn menu start, nhưng menu chính sẽ vẫn hiển thị.	Ứng dụng phủ toàn bộ vùng desktop

2.5.4 Thuộc tính Size

Thuộc tính Size xác định kích thước của cửa sổ ứng dụng. Phụ thuộc vào giá trị của thuộc tính `FormBorderStyle`, ứng dụng có thể bỏ qua giá trị thuộc tính Size hoặc thiết lập giá trị kích thước đặc biệt cho ứng dụng. Trên Pocket PC

2.5.5 Thiết lập vị trí của Form bằng thuộc tính Location

Thuộc tính `Location` xác định góc trên bên trái của Form. Trên Pocket PC thuộc tính `Location` không có ảnh hưởng trừ khi thuộc tính `FormBorderSytle` được thiết lập là `FormBorderSytle.None`. Trên Windows CE vị trí của cửa sổ luôn luôn bằng thuộc tính `Location`, trừ khi ứng dụng đưa vào trạng thái phóng to hoặc thu nhỏ hết cỡ.

2.6 Điều khiển `Button`

Lớp `System.Windows.Forms.Button` được .NET bổ sung một điều khiển button. Khi người sử dụng bấm vào nút lệnh. Chúng ta có thể thao tác sự kiện này bằng sự thực thi `System.EventHandler`. Đoạn mã sau đây là sự thực thi `EventHandler` cái đó hiển thị thời gian hiện hành.

```
Private void button_Click(object sender, System.EventArgs e) {
  MessageBox.Show(DateTime.Now.ToShortTimeString(),
                 "The Current Time Is",
                 MessageBoxButtons.OK,
                 MessageBoxIcon.Exclamation,
                 MessageBoxDefaultButton.Button1);
}
```

Hình 2.4. Ứng dụng `GiveEmTime` thực thi trên Pocket PC 2002 emulator. Nút có nhãn **What is the Time** đã được bấm, và thời gian hiện hành được hiển thị trong hộp thoại.

Hình 2.4. Ứng dụng `GiveEmTime` chạy trên Pocket PC 2002 emulator.



Bảng 2.3. Mã phím được phát sinh bằng Directional Pad trên thiết bị Pocket PC

Giá trị KeyCode	Nút phần cứng liên quan
`Keys.Up`	Nút trên được bấm
`Keys.Down`	Nút dưới được bấm
`Keys.Left`	Nút bên trái được bấm
`Keys.Right`	Nút bên phải được bấm
`Keys.Return`	Nút giữa được bấm

2.7 Điều khiển `TextBox`

Điều khiển cho phép người dùng có thể nhập dữ liệu đầu vào cho ứng. Điều khiển `TextBox` hỗ trợ thuộc tính `BackColor` và `ForeColor`, không giống như hầu hết các điều khiển khác trong .NET Compact Framework. Sự kiện Click không hỗ trợ, nhưng có hỗ trợ các sự kiện `KeyPress`, `KeyUp`, và `KeyDown`. Thuộc tính `PasswordChar` được hỗ trợ.

2.8 Điều khiển `Label`

Điều khiển nhãn cho phép chúng ta hiển thị văn bản tới người sử dụng. Thuộc tính Text của điều khiển xác định văn bản sẽ được hiển thị tới người sử dụng. Văn bản hiển thị có thể có sự căn lề khác nhau dựa vào thuộc tính `TextAlign`. Thuộc tính `TextAlign` có thể nhận các giá trị là `TopLeft`, `TopCenter`, và `TopRight`.

2.9 Điều khiển `RadioButton`

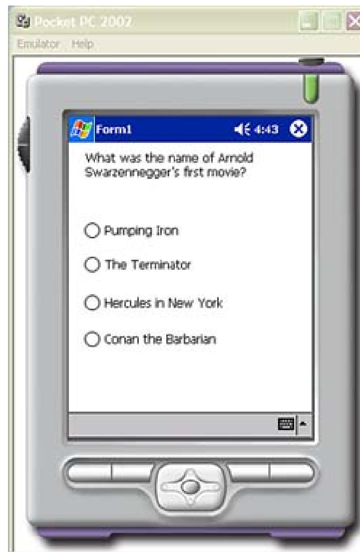
Nút điều khiển Radio đưa tới người sử dụng một dãy các giá trị lựa chọn loại trừ nhau. Khi một nút radio trong một nhóm được chọn, các nút khác sẽ tự động bị bỏ chọn.

Lớp `RadioButton` có hai sự kiện được đưa ra khi trang thái chọn của `RadioButton` thay đổi: `Click` và `CheckedChanged`. Sự kiện `Click` phát sinh khi người sử dụng chọn vào nút radio. Chúng ta có thể thao tác với sự kiện này như là đối với sự kiện Click của lớp button. Sự kiện `CheckedChanged` được phát sinh khi trạng thái chọn của `RadioButton` thay đổi bằng lập trình hay giao diện đồ hoạ.

Sự kiện `Click` sẽ không phát sinh nếu thuộc tính `Checked` của `RadioButton` được thay đổi bằng lập trình.

Ứng dụng demo `Arnie.exe`, làm thế nào để sử dụng một nhóm các điều khiển. Hình 2.5 cho thấy ứng dụng chạy trên Pocket PC emulator.

Hình 2.5. Ứng dụng Arnie chạy trên Pocket PC 2002 emulator.



Sau đây là đoạn mã demo thao tác với sự kiện `CheckedChanged`.

```
private void radioButton2_CheckedChanged(object sender,
        System.EventArgs e) {
  if(this.radioButton2.Checked)
    MessageBox.Show
      ("Wrong, The Terminator (1984) O.J Simpson almost got the
role...",
               "Wrong!");
}
```

2.10 Điều khiển CheckBox

Điều khiển `CheckBox` giống như điều khiển `RadioButton`. Điều khiển này đưa đến cho người sử dụng danh sách các lựa chọn. Điều khác là điều khiển `CheckBox` có thể có nhiều lựa chọn trong cùng một lúc, trong khi điều khiển `RadioButton` lựa chọn loại trừ.

Điều khiển `CheckBox` cung cấp thuộc tính `CheckState`, xác đinh điều khiển nào được chọn. Thuộc tính `CheckState` thực chất là một bảng liệt kê. Thành phần của nó là `Unchecked`, `Checked`, và `Indeterminate`. Trạng thái `Indeterminate` chỉ có thể được sử dụng khi thuộc tính `ThreeState` của điều khiển `CheckBox` được thiết lập là true. Khi `CheckState` là `Indeterminate` và thuộc tính `ThreeState` là true, điều khiển được khoanh thành ô vuông. Có nghĩa là trạng thái chọn không thể kiểm soát. Điều khiển sẽ không trả kết quả tới người sử dụng khi chọn trong suất quá trình thuộc tính `AutoCheck` được thiết lập là `false`. Khi thuộc tính `AutoCheck` được thiết lập `true`, khi đó có thể bấm chọn trên điều khiển.

Ứng dụng `Apples.exe` là một ví dụ khác đơn giản là xác định loại táo người sử dụng thích. Điều khiển `CheckBox` trên cùng có nhãn là "I like apples.". Các điều khiển `CheckBox` khác có nhãn cùng với loại táo khác nhau và một trạng thái mờ mờ cho đến khi `CheckBox` có nhãn "I like apples" được chọn, khi đó người sử dụng lựa chọn loại táo anh ta hoặc cô ta thích. Hình 2.6 cho chúng ta thấy ứng dụng chạy trên Pocket PC emulator.

Hình 2.6. Các trạng thái của điều khiển `CheckBox` chạy trên Pocket PC 2002.

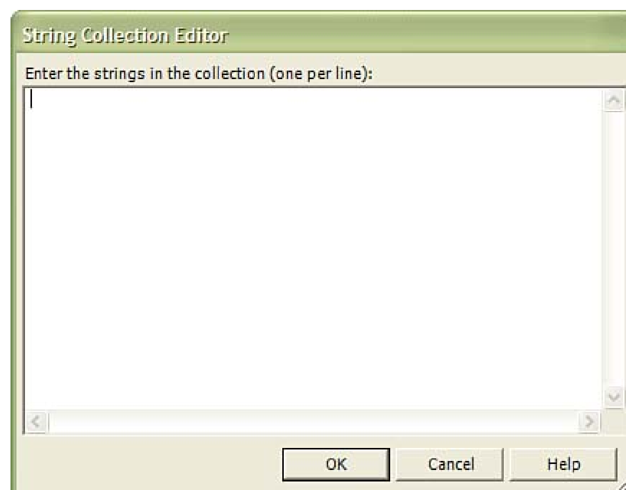


2.11 Điều khiển ComboBox

Điều khiển `ComboBox` là điều khiển thể hiện một danh sách các lựa chọn trong sự hạn chế của màn hình. `ComboBox` xuất hiện như là điều khiển `TextBox` cùng với một mũi tên bên tay phải. Một danh sách lựa chọn thả xuống dưới điều khiển khi người sử dụng chọn vào mũi tên. Khi người sử dụng lựa chọn một tùy chọn hoặc chọn lại mũi tên, danh sách các tuỳ chọn sẽ cuộn lên.

Để thêm một mục vào điều khiển `ComboBox` có thể hoàn thành lúc thiết kế và lúc thực thi. Để thêm một mục vào `ComboBox` lúc thiết kế, đơn giản là chọn `ComboBox` trong Form Designer. Sau đó chọn vào phần bên phải tên thuộc tính Items trong cửa sổ thuộc tính. Nó sẽ đưa đến một hộp thoại String Collection Editor (sem hình 2.7). Trong hộp thoại String Collection Editor, đưa vào danh sách các mục sẽ xuất hiện trong `ComboBox`. Mỗi mục phải xuất hiện trên cùng một dòng.

Hình 2.7. Hộp thoại String Collection Editor.



Các mục có thể được thêm vào điều khiển ComboBox lúc thực thi. Điều này có thể hoàn thành bằng hai cách:

Cách 1: Gọi phương thức Add trên thuộc tính tập hợp Items của điều khiển ComboBox. Các mục có thể loại bỏ thông qua phương thứ Remove trên tập hợp Items, hoặc tất cả các mục có thể loại bỏ bằng cách gọi phương thức Clear. Đoạn mã sau thêm ba chuỗi vào điều khiển ComboBox có tên comboBox1

```
comboBox1.Items.Add("Hi");
comboBox1.Items.Add("Howdy");
comboBox1.Items.Add("Wuz Up");
```

Cách 2: Chúng ta có thể thêm vào ComboBox lúc thực thi bằng cách ràng buộc điều khiển với một đối tượng tập hợp. Điều này được hoàn thành bằng cách thiết lập DataSource với một đối tượng tập hợp. Khi ComboBox cố gắng thêm một mục vào danh sách, nó sẽ gọi phương thức ToString trên mỗi mục trong DataSource và thêm vào danh sách lựa chọn. Chuỗi có thể tuỳ biến bằng cách thiết lập thuộc tính DisplayName của điều khiển ComboBox. ComboBox sẽ gọi thuộc tính riêng biệt trong thuộc tính DisplayName và thêm chuỗi trả về vào danh sách lựa chọn.

Đoạn mã Listing 2.1 mô tả cách ràng buộc một ComboBox với một danh sách đối tượng tuỳ biến. Lớp Customer là một lớp tuỳ biến lưu trữ tên của khách hàng. Lớp có một thuộc tính FullName, thuộc tính này lưu trữ tên đầy đủ. Khi ComboBox được giới hạn trong phương thức LoadCustomer, thuộc tính FullName được thiết lập như là DisplayName.

Listing 2.1

```
class Customer {
  string m_First;
  string m_Middle;
  string m_Last;

  public Customer(string first, string middle, string last) {
    m_First = (first == null) ? string.Empty : first;
    m_Middle = (middle == null) ? string.Empty : middle;
    m_Last = (last == null) ? string.Empty : last;
  }

  public string FirstName {
    get { return m_First; }
  }

  public string MiddleName {
    get { return m_Middle; }
  }

  public string LastName {
    get { return m_Last; }
  }

  static string FullNameWithInitial = "{0} {1}. {2}";
  static string FullNameNoInitial = "{0} {1}";
  public string FullName {
    get {
      return (m_Middle.Length > 0) ?
```

```
        string.Format(FullNameWithInitial, m_First, m_Middle[0], m_Last) :
        string.Format(FullNameNoInitial, m_First, m_Last);
    }
  }
}

private void LoadCustomers() {
  if(customers != null)
    return;

    customers = new Customer[6];
    customers[0] = new Customer("Ronnie", "Donnell", "Yates");
    customers[1] = new Customer("Moya", "Alicia", "Hines");
    customers[2] = new Customer("Veronica", "Christine", "Yates");
    customers[3] = new Customer("Diane", "", "Taylor");
    customers[4] = new Customer("Kindell", "Elisha", "Yates");
    customers[5] = new Customer("Zion", "Donnell", "Yates");

    this.comboBox1.DataSource = customers;
    this.comboBox1.DisplayMember = "FullName";
}
```

Có hai cách để lấy mục đang được chọn trong điều khiển `ComboBox`. Thứ nhất, thuộc tính `SelectedIndex` trả về chỉ số của mục đang chọn. Chỉ số này có thể được sử dụng để truy cập mục đang chọn từ thuộc tính `Items` của điều khiển `ComboBox`. Đoạn mã sau minh hoạ thuộc tính `SelectIndex`:

```
string selItem = comboBox1.Items[comboBox1.SelectedIndex].ToString();
```

Điều khiển `ComboBox` cung cấp thuộc tính `SelectedItem`, thuộc tính này trả về một tham chiếu đến mục đang chọn. Một là chúng ta có thể tham chiếu đến mục đang chọn, chúng ta không cần phải đưa chỉ số vào thuộc tính `Items` . Đoạn mã sau mô tả cách sử dụng thuộc tính `SelectedItem`:

```
string selItem = comboBox1.SelectedItem.ToString();
```

2.12 Điều khiển ListBox

`ListBox` sẽ được sử dụng nếu chúng ta có đủ không gian màn hình để hiển thị một vài tuỳ chọn cho người sử dụng trong một lần.

`ComboBox` và `ListBox` có các thuộc tính và các phương thức giống nhau. Bao gồm thuộc tính tập hợp `Items` và các thương thức `Add`, `Remove`, và `Clear` trên thuộc tính `Items` . Ví dụ, đoạn mã sau thêm chuỗi vào điều khiển `ListBox` lúc thiết kế.

```
listBox1.Items.Add("Hi");

listBox1.Items.Add("Howdy");

listBox1.Items.Add("Wuz Up");
```

Chúng ta có thể thêm vào điều khiển `ListBox` lúc thực thi bằng cách gắn `ListBox` với một tập hợp. Trong quá trình gắn một điều khiển `ListBox` giống với quá trình trong điều khiển `ComboBox`. Trước tiên, thiết lập `DataSource` với một tập hợp. Sau đó, thiết lập thuộc tính `DisplayMember` với một mục trong nguồn dữ liệu, mục này sẽ được hiển thị như là một chuỗi.

```
private void LoadCustomers() {
```

```
    if(customers != null)
      return;

      customers = new Customer[6];
      customers[0] = new Customer("Ronnie", "Donnell", "Yates");
      customers[1] = new Customer("Moya", "Alicia", "Hines");
      customers[2] = new Customer("Veronica", "Christine", "Yates");
      customers[3] = new Customer("Diane", "", "Taylor");
      customers[4] = new Customer("Kindell", "Elisha", "Yates");
      customers[5] = new Customer("Zion", "Donnell", "Yates");

      this.listBox1.DataSource = customers;
      this.listBox1.DisplayMember = "FullName";
  }
```

`ListBox` có hai thuộc tính `SelectedIndex` và `SelectedItem` cho phép truy cập mục đang chọn.

2.13 Các điều khiển khác

- `NumericUpDown`
- `DomainUpDown`
- `ProgressBar`
- `StatusBar`
- `TrackBar`
- `ToolBar`
- `MainMenu`
- `ContextMenu`
- `Timer`
- `OpenFileDialog` và `SaveFileDialog`
- `Panel`
- `HScrollBar` và `VScrollBar`
- `ImageList`
- `PictureBox`
- `ListView`
- `TabControl`
- `TreeView`
- `DataGrid`

Chương 3 Khả năng kết nối mạng bằng .Net Compact Framework

3.1 Sockets

Socket là chuẩn cho truyền thông với các máy tính trên mạng cục bộ (LAN) và mạng diện rộng (WAN), giống như là Internet. Hai máy tính giao tiếp với mỗi máy khác bằng cách sử dụng socket, sau đó nó trở thành giao thức phổ biến khi mà một máy tính đang mong chờ kết nối để nhận một kết nối, và một máy khác tạo kết nối khởi tạo.

- Máy tính mong chờ nhận một kết nối, host hoặc server, lắng nghe kết nối vào trên một cổng nào đó. Máy tính có một địa xhỉ IP duy nhất, giống như là 172.68.112.34, và hàng nghìn cổng sẵn sàng, nó sẵn sang cho nhiều chương trình cùng lắng nghe kết nối, mỗi kết nối sử dụng một cổng riêng.

- Máy tính tạo ra khởi tạo kết nối (client), xác định địa chỉ IP của máy mong chờ kết nối (server). Nếu biết được tên của máy mong chờ kết nối như là `www.mycomputer.org`, chúng ta có thể sử dụng DNS tra cứu để xác định địa chỉ IP liên quan đến tên.

- Client quyết định cổng nào kết nối với host. Ví dụ: Web servers luôn luôn lắng nghe trên cổng 80, vì vậy máy tính muốn kết nối với máy Web server khác quá trình luôn biết nó cần thiết kết nối trên cổng 80. Ứng dụng thường sử dụng một lượng lớn các cổng không giống nhau, được sử dụng bởi bất kỳ ai, như là 10998. Phạm vi số hiệu cổng mà ứng dụng có thể sử dụng phụ thuộc vào hệ điều hành. Một số hệ điều hành dự trữ một số số hiệu cổng đặc biệt, ví dụ 1024. Để an toàn nên chọn các cổng từ 2000 và 60000.

- Client có thể kết nối tới địa chỉ IP và số hiệu cổng. Host nhận kết nối. Khi đó tồn tại một kết nối socket giữa hai máy tính.

- Client và host gửi các gói dữ liệu qua lại.

Trong phần này chúng ta học cách thao tác kết nối socket bằng .NET Compact Framework.

3.1.1 Giao thức: TCP/IP, UDP

Tổng quan, lập trình socket sử dụng giao thức Internet để gửi các gói tin giữa hai máy. Có hai kiểu gói tin sử dụng để gửi dữ liệu thông qua giao thức Internet:

Gói tin TCP:

Đây là kiểu gói tin thường được sử dụng trên Internet để truyền dữ liệu đi xa, giao thức của gói tin TCP trên giao thức Internet gọi là mạng TCP/IP. Nếu một máy tính gửi một gói tin TCP qua một kết nối Socket, dữ liệu trong gói đó được bảo đảm tới đích mà không có lỗi. Nếu gói tin tới đích nhưng có lỗi, sau đó dữ liệu lại được gửi lại. Nếu gói tin không tới đích trong khoảng thời gian cho phép, sau chức năng thường được gọi để gửi báo báo gói tin có lỗi. Cách

kiểm tra lỗi thay đổi tuỳ theo từng nền tảng (platform), nhưng chúng ta sẽ nghiên cứu quá trình xử lý này chi tiết cho .NET Compact Framework.

Gói tin UDP

Gói tin này khác với gói tin TCP, bởi vì nó không đảm bảo gói UDP sẽ tới đích hoặc dữ liệu sẽ không có lỗi. Tuy nhiên, sự thiếu đi quá trình kiểm tra lỗi có nghĩa là sử dụng gói tin UDP làm cho phần đầu của gói tin nhỏ hơn, vì vậy chương trình có thể truyền dữ liệu nhanh hơn. Một ứng dụng tốt sử dụng gói tin UDP là điện thoại Internet

3.1.2 Sự thực thi của IP: IPv4 hay IPv6

Quá trình xử lý của kết nối máy khách tới máy chủ bao gồm xác định địa chỉ IP của máy chủ và sau đó tạo kết nối. Sự phức tạp của quá trình truyền đi và truyền lại đúng địa chỉ là trách nhiệm của giao thức Internet. Giao thức này có một vài phiên bản. Giao thức Internet phiên bản 4, IP4 là phổ biến nhất được sử dụng trên Internet. Một địa chỉ IPv4 bao gồm bốn phần 8 bít. Một địa chỉ IPv4 gồm bốn phần, mỗt phần bao gồm các số thập phân từ 0 đến 255, các phần được cách nhau bởi dấu “.”, giống như là `172.68.112.34`.

Ngày nay để kết nối với thế giới, IPv4 không cung cấp đủ địa chỉ duy nhất cho mỗi máy tính.

Phiên bản mới nhất của giao thức IP là 6, thường viết là IPv6. Nó không được sử dụng phổ biến. IPv6 bao gồm tăng cường tính bảo mật và địa chỉ. IPv6 sẽ cung cấp đủ địa chỉ IP duy nhất cho mỗi máy tính trong tương lai.

.NET Compact Framework hỗ trợ nhiều hơn cho phiên bản trước (IPv4). Nó không hỗ trợ giao thức IPv6. Trong phần này chúng ta chỉ tìm hiểu về giao thức IPv4.

3.2 Lập trình `Socket` với .NET Compact Framework

Lớp `System.Net.Sockets.Socket`. Thủ tục để nhận một lớp Socket kết nối với máy ở xa phụ thuộc vào máy tính đó, tuy nhiên quá trình xử lý để đọc và ghi dữ liệu là giống nhau.

Để sử dụng các lớp xử lý mạng trong .NET Compact Framework, chúng ta phải khai báo không gian tên System.Net. Ví dụ: using System.Net.

3.2.1 Tạo kết nối từ máy khách tới máy chủ (client)

Để tạo một kết nối thành công, trước tiên chúng ta phải tìm hiểu lớp `System.Net.EndPoint`. Để lưu giữ thông tin về điểm cuối nơi mà kết nối đến: địa chỉ IP của máy chủ và số hiệu cổng mong muốn. Để thiết lập đúng điểm cuối và sử dụng nó để kết nối socket tới máy chủ, chúng ta làm theo các bước sau:

Bước 1: Khai báo biến điểm cuối (EndPoint) và biến Socket.

Bước 2: Điểm cuối gồm thông tin địa chỉ và số hiệu cổng. Có hai cách để làm điều này, phụ thuộc vào địa chỉ của máy chủ, giống như là: `172.68.25.34`, hoặc tên DSN của máy chủ, như là `www.mycomputer.net`.

Tìm địa chỉ IP của một máy chủ:

Nếu chúng ta biết địa chỉ IP của máy chủ, sử dụng `IPAddress` trong cấu trúc. Ví dụ sau mô tả khởi tạo một điểm cuối, máy chủ có địa chỉ IP là `172.68.25.34`, và cổng 9981:

```
EndPoint l_EndPoint = new IPEndPoint( IPAddress.Parse(
        "172.68.25.34"), Convert.ToInt16(9981));
```

Nếu chúng ta không biết địa chỉ IP, chúng ta phải dùng DSN để tìm địa chỉ IP của máy chủ thông qua tên. DSN tìm kiếm trả lại địa chỉ IP tương ứng với tên. Đoạn mã sau là một trường hợp:

```
IPHostEntry l_IPHostEntry = Dns.Resolve("www.mycomputer.net");
EndPoint l_EndPoint = new IPEndpoint(l_IPHostEntry.AddressList[0],
9981);
```

Bước 3: Sử dụng điểm cuối (EndPoint) để thử kết nối socket tới máy chủ. Chúng ta phải sử dụng mệnh đề try/catch ở đây, bởi vì thử kết nối sẽ đưa ra một ngoại lệ nếu có vấn đề, như máy chủ từ chối không chấp nhận kết nối hoặc máy chủ không tồn tại,...

Ví dụ sau mô tả ba bước ở trên:

```
try
{
    Socket l_Socket = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream, ProtocolType.Tcp);
    l_Socket.Connect(l_EndPoint);
    if (l_Socket.Connected){
       // l_Socket bây giờ có thể gửi và nhận dữ liệu
    }
}
catch (SocketException e)
{ /* Đưa ra thông báo lỗi,… */ }
```

3.2.2 Tạo kết nối từ máy chủ lằng nghe từ máy khách (Host)

Chúng ta có thể thu được một kết nối socket từ máy tính ở xa bằng cách đảm nhiệm như là máy chủ. Khi một thiết bị như máy chủ, nó đợi nhận kết nối từ các máy khách. Để tạo kết nối để thiết bị của chúng ta như là máy chủ, chúng ta phải thiết lập một socket lắng nghe trên một cổng đến khi một ai đó gửi một yêu câu kết nối đến thiết bị của chúng ta. Sau đây là các bước tạo socket lắng nghe trên một cổng để cho máy khác kết nối tới:

Bước 1: Tạo một socket để lắng nghe kết nối.

Bước 2: Ràng buộc socket lắng nghe trên một cổng. Nó chỉ lắng nghe kết nối trên một cổng.

Bước 3: Gọi `Accept()` trên socket lắng nghe nhận được từ socket khác khi một ai đó kết nối tới. Đoạn mã có thể đọc và ghi socket nhận được, và socket tiếp tục đợi kết nối mới.

Ví dụ sau mô tả ba bước ở trên:

```
m_listenSocket = new Socket(AddressFamily.InterNetwork,
        SocketType.Stream, ProtocolType.Tcp);

m_listenSocket.Bind(new IPEndPoint(IPAddress.Any, 8758));
m_listenSocket.Listen((int)SocketOptionName.MaxConnections);
```

```
m_connectedSocket = m_listenSocket.Accept();
if (m_connectedSocket != null)
{
    if (m_connectedSocket.Connected)
    {
       // Someone has connected to us.
    }
 }
```

3.2.3 Gửi và nhận trên Socket đã kết nối

Một socket được kết nối tới máy tính ở xa. Nó có thể sử dụng gửi và nhận dữ liệu. Cách đơn giản nhất để làm việc này là gọi `Socket.Send()` để gửi dữ liệu và `Socket.Receive()` nhận dữ liệu.

3.2.3.1 Gửi dữ liệu vào một Socket cùng với Socket.Send

`Socket.Send()` có bốn thành phần nạp chồng, mỗi thành phần là một mức khác nhau của điều khiển thông qua cái được gửi:

- `Send(Byte[] buffer)`: Gửi tất cả mọi thứ bên trong mảng byte buffer.

- `Send(Byte[] buffer, SocketFlags socketFlags)` Gửi tất cả mọi thứ trong buffer cùng với sự hạn chế riêng thông qua cách dữ liệu đi như thế nào.

- `Send(Byte[] buffer, Int32 size, SocketFlags socketFlags)`: Gửi tất cả dữ liệu trong buffer tuỳ theo kích cỡ `size`. Nếu chúng ta muốn gửi chỉ một phần của một buffer, sau đó có thể chỉ rõ `SocketFlags.None` sử dụng mặc định hành vi gửi. Ví dụ, để gửi 16 byte đầu tiền của mảng, chúng ta có thể sử dụng `l_Socket.Send(l_buffer, 16, SocketFlags.None)`.

- `Send(Byte[] buffer, Int32 offset Int32 size, SocketFlags socketFlags)`: Giống như thành phần trên chỉ khác là chúng ta có thể chỉ rõ chỉ số bắt đầu của mảng. Ví dụ, để gửi từ byte tứ 3 đến bute thứ 7 của mảng, chúng ta có thể sử dụng như sau:

```
l_Socket.Send(l_buffer, 2, 6, SocketFlags.None);
```

Phương thức Send trả về số byte gửi thành công. Vấn đề này cùng với phương thức `send()` dường như giống nhau rất nhiều việc biến đổi tất cả mọi cái chúng ta muốn gửi vào mảng các byte để gửi thông qua socket. .NET Compact Framework hỗ trợ hai lớp rất hữu ích, `System.Text.Encoding` và `System.Convert`, hai lớp này giúp chuyển đổi kiểu dữ liệu cơ bản thành mảng các byte để có thể gửi qua socket.

Cách dễ nhất để tìm hiểu cách sử dụng lớp `Encoding` và `Convert` là xem ví dụ. Sau đây là ví dụ socket có tên là `l_Socket` đã tồn tại và đã được kết nối:

- Gửi một chuỗi sử dụng mã hoá ASCII :

```
l_Socket.Send(Encoding.ASCII.GetBytes("Send me")
```

- Gửi một chuỗi sử dụng mã hoá Unicode:

```
l_Socket.Send(Encoding.Unicode.GetBytes("Send me")
```

- Gửi một số nguyên có giá trị 2003:

```
l_Socket.Send(Encoding.ASCII.GetBytes(Convert.ToString(2003))
```

- Gửi một số thực có giá trị 2.7:

```
l_Socket.Send(Encoding.ASCII.GetBytes(Convert.ToString(2.71))
```

3.2.3.2 Nhận dữ liệu từ từ socket bằng `Socket.Receive`

Nhận dữ liệu từ một socket thông qua phương thức `Socket.Receive`. Receive có bốn thành phần nạp chồng, giống như thành phần nạp chồng của `Socket.Send`. Mỗi thành phần nạp chồng trả về số byte đọc thành công:

- `Receive (Byte[] buffer)`: Thành phần này nhận dữ liệu trong bộ đệm.

- `Receive (Byte[] buffer, SocketFlags socketFlags)` Thành phần này nhận dữ liệu trong bộ đệm bằng cách sử dụng cờ để chỉ ra dữ liệu được lấy như thế nào.

- `Receive (Byte[] buffer, Int32 size, SocketFlags socketFlags)` Thành phần này nhận tuỳ theo kích cữ của dữ liệu trong bộ đệm. Nếu dữ liệu nhiều hơn dữ liệu sẵn sàng, nó được bỏ qua. Chúng ta có thể nhận dữ liệu còn lại bằng cách gọi lại `Receive`. Nếu chúng ta chỉ muốn nhận những byte mà chúng ta không nhận được, sau đó chúng ta có thể chỉ `SocketFlags.None` để sử dụng mặc định cho hành động gửi. Ví dụ để nhận 16 byte đầu tiên của dữ liệu sẵn sàng, sử dụng `l_Socket.Receive(l_buffer, 16, SocketFlags.None)`

- `Receive (Byte[] buffer, Int32 offset Int32 size, SocketFlags socketFlags)` Thành phần này giống như thành phần trước, chỉ khác là chúng ta có thể chỉ ra chỉ số trong mảng để sử dụng bắt đầu ghi dữ liệu vào mảng. Ví dụ, để nhận 7 byte dữ liệu trong bộ đệm bắt đầu từ vị trí thứ 3 trong bộ đệm, sử dụng đoạn mã sau:

```
l_Socket.Receive(l_buffer, 2, 6, SocketFlags.None);
```

Có kỹ thuật cho phép chuyển đổi dữ liệu để gửi từ socket ra mảng, kỹ thuật đơn giản nhất là chuyển đổi mảng byte trong kiểu dữ liệu cơ bản. Như phần trước, lớp `Encoding` và `Convert` cung cấp phương tiện cho chuyển đổi, và chúng ta sẽ xem trong ví dụ. Đây là ví dụ thừa nhận dữ liệu đã được nhận trong mảng Byte có tên là `l_Buffer`:

- Chuyển đổi các byte nhận được trong một chuỗi ASCII :

```
string l_ASCII = Encoding.ASCII.GetString(l_Buffer);
```

- Chuyển đổi các nhận được trong một chuỗi Unicode:

```
string l_ASCII = Encoding.Unicode.GetString(l_Buffer);
```

- Chuyển đổi các byte nhận được, cái đó là mã ASCII text integer:

```
int l_Integer = Convert.ToInt32(Encoding.ASCII.GetString(l_Buffer));
```

- Chuyển đổi các byte nhận được, cái đó là mã ASCII text integer, into a Double:

```
Double l_Double = Convert.ToInt32(Encoding.ASCII.GetString(l_Double));
```

Bảng 3.1. Danh sách các thành phần chuyển đổi được hỗ trợ bởi lớp `Convert` trên .NET Compact Framework.

Bảng 3.1. Lớp `Convert` trên .NET Compact Framework

Phương thức	Tên của các kiểu dữ liệu đầu vào được chấp nhận
ToBoolean	object, bool, sbyte, char, byte, short, ushort, int, uint, long, String, float, double, decimal
ToChar	object, char, sbyte, byte, short, ushort, int, uint, long, ulong, String, float, double, decimal
ToSByte	object, bool, sbyte, char, byte, short, ushort, int, uint, long, ulong, float, double, decimal, String
ToByte	object, bool, byte, char, sbyte, short, ushort, int, uint, long, ulong, float, double, decimal, String
ToInt16	object, bool, char, sbyte, byte, ushort, int, uint, short, long, ulong, float, double, decimal, String
ToUInt16	object, bool, char, sbyte, byte, short, int, ushort, uint, long, ulong, float, double, decimal, String
ToInt32	object, bool, char, sbyte, byte, short, ushort, uint, int, long, ulong, float, double, decimal, String
ToUInt32	object, bool, char, sbyte, byte, short, ushort, int, uint, long, ulong, float, double, decimal, String
ToInt64	object, bool, char, sbyte, byte, short, ushort, int, uint, ulong, long, float, double, decimal, String
ToUInt64	object, bool, char, sbyte, byte, short, ushort, int, uint, long, UInt64, float, double, decimal, String
ToSingle	object, sbyte, byte, char, short, ushort, int, uint, long, ulong, float, double, decimal, String, bool
ToDouble	object, sbyte, byte, short, char, ushort, int, uint, long, ulong, float, double, decimal, String, bool
ToDecimal	object, sbyte, byte, char, short, ushort, int, uint, long, ulong, float, double, String, decimal, bool, DateTime
ToDateTime	object, String
ToString	Object, bool, char, sbyte, byte, short, ushort, int, uint, long, ulong, float, double, decimal, Decimal, DateTime
ToBase64String	byte[]
byte[]FromBase64 String	String
ToBase64CharArra y	byte[]
byte[]FromInt64C harArray	char[]

3.3 Tuần tự hóa đối tượng để truyền qua Socket

Phiển bản desktop của .NET Framework cho phép tuần tự hóa hầu hết kiểu đối tượng thành mảng các byte, để có thể gửi qua socket. Các đối tượng phức tạp, người phát triển thực thi giao diện `ISerializable`, cùng với mã tuần tự (serialize) và hồi phục (deserialize) đối tượng dữ liệu.

.NET Compact Framework không hỗ trợ những chức năng này. Lớp `DataSet` là lớp duy nhất có thể tự tuần tự hóa. Thông thường lớp `DataSet` được sử dụng như là một cơ sở dữ liệu quan hệ trong bộ nhớ. Nó là một ý tưởng cho bộ đệm dữ liệu nhỏ dựa vào máy chủ ở xa trong khi duy trì cấu trúc quan hệ của dữ liệu. `DataSet` có thể lưu trữ tất cả các kiểu dữ liệu cơ bản trên .NET Compact Framework.

3.4 Sử dụng gói UDP

Như đã đề cập, có hai kiểu gói tin thường được sử dụng để truyền tin trên mạng. Kiểu chung nhất, gói TCP phải chọn cho gần như tất cả các trường hợp bởi vì nó đảm bảo rằng dữ liệu đến không bị hư hỏng hoặc ngược lại trả lại tín hiệu lỗi nếu có một vấn đề gì mà không thể sửa chữa.

Gói tin UDP rất hữu ích cho các ứng dụng dòng thời gian thực.

Gói tin UDP khác gói tin TCP trong cách mà chúng kết nối, giao thức TCP là giao thức hướng kết nối, điều này có nghĩa là chúng ta cần kết nối với một socket trên máy tính ở xa trước khi chúng ta có thể gửi hoặc nhận dữ liệu bằng socket. Giao thức kết nối không yêu cầu bất kỳ kết nối nào được thiết lập trước khi có gắn gửi hoặc nhận dữ liệu. Nếu không có một lắng nghe trên địa chỉ IP và cổng nơi mà gói UDP được gửi, sau đó gói tin đơn giản là bị mất.

Cách đơn giản nhất đển làm việc với gói UDP là sử dụng lớp `UdpClient`, lớp này được .NET Compact Framework hỗ trợ. Lớp `UdpClient` cho phép các lập trình viên gửi các byte tới nhóm ở xa. `UdpClient` cho phép người phát triển nhận byte từ nhóm ở xa hoặc từ bất kỳ người nào cố gắng gửi dữ liệu tới cổng mà `UdpClient` lắng nghe. Quan tâm đến các cấu trúc và phương thức được `UdpClient` sử dụng sau:

- `void Connect(String hostname, Int32 port)` Thiết lập kết nối tới một máy tính có địa chỉ IP tương ứng được chỉ ra bởi tên máy chủ (hostname) và số hiệu cổng (port). Sau đó sử dụng phương thức `Send(Byte[] dgram, Int32 bytes)` sẽ gửi dữ liệu đến vị trí được chỉ ra trong phân kết nối. Phương thức này trả về kiểu `void` bởi vì không có khái niệm kết nối thành công khi sử dụng gói UDP. Phương thức này chỉ đơn thuần là tạo để gửi dữ liệu tới một địa chỉ IP và số hiệu cổng.

- `void Connect(IPAddress addr, Int32 port)` Giống như phương thức trước, ngoại trừ cho phép bạn chỉ ra máy tính ở xa bằng `IPAddress` và `port`. Đoạn mã ví dụ:

```
l_UdpClient.Connect(IPAddress.Parse("172.68.25.34"), 9981)
```

- `void Connect(IPEndpoint endPoint)` Kết nối với máy ở xa bằng cách chỉ ra `endPoint`.

- `Int32 Send(Byte[] dgram, Int32 bytes, IPEndPoint endPoint)` Gửi tất cả `bytes` của bộ đệm `dgram` tới máy tính có địa chỉ IP và cổng được chỉ ra trong `endPoint`.

- `Send(Byte[] dgram, Int32 bytes, String hostname, Int32 port)` Gửi tất cả các `bytes` của bộ đệm `dgram` tới máy tính có địa chỉ IP tương ứng với `hostname` và cổng, như trong đoạn mã ví dụ sau:

```
Send(aBuffer, aBuffer.Length, "www.mycomputer.net", 9981)
```

Phương thức trên trả về số byte gửi.

- Send(Byte[] dgram, Int32 bytes) Gửi tổng số byte của bộ đệm tới máy chủ ở xa đươc chỉ ra trong phương thức kết nối. Để sử dụng thành phần nạp chồng, chúng ta trước tiền phải gọi Connect, vì vậy UdpClient biết nơi gửi gói UDP. Phương thức này trả về số byte gửi được.

- Receive(ref IPEndPoint remoteEP) Đợi để nhận dữ liệu từ EndPoint. Chúng ta có thể tạo một EndPoint tham chiếu đến một địa chỉ IP và cổng, hoặc chúng ta có thể thiết lập EndPoint để nhận dữ liệu từ bất kỳ địa chỉ IP và port. EndPoint được cập nhật sau khi dữ liệu được nhận cho biết nơi dữ liệu đến.

Viết mã cho UdpClient

Đoạn mã này mô tả cách thiết lập một UdpClient, sau đó gửi gói tin UDP tới máy tính có địa chỉ IP là 192.168.0.200, cổng 8758. Chú ý là thông qua gọi phương thức UdpClient.Connect(). UdpClient biết nơi gửi gói tin UDP khi UdpClient.Send() được gọi, nhưng không có kết nối liên tục.

```
IPEndPoint senderIP = new
        IPEndPoint(IPAddress.Parse("192.168.0.200"),
        Convert.ToInt32(8758));

UdpClient l_UdpClient = new UdpClient();
l_UdpClient.Connect(senderIP);

for (int i = 0; i < 20; i++)
{
    l_UdpClient.Send(Encoding.ASCII.GetBytes("Hello_UDP_1"),
            Encoding.ASCII.GetBytes("Hello_UDP_1").Length);

    System.Threading.Thread.Sleep(1000);
}

l_UdpClient.Close();
```

Sau đây đoạn mã tạo một vòng lặp. Mỗi lần lặp của khối lặp và lắng nghe trên cổng 8758 đến khi nó nhận một gói tin từ bất kỳ địa chỉ IP nào.

```
IPEndPoint listenerIP = new IPEndPoint(IPAddress.Any, 8758);
UdpClient listener = new UdpClient(listenerIP);

for (int i = 0; i < Convert.ToInt16(this.txtMaxPackets.Text); i++)
{
    // Now receive the three datagrams from the listener
    IPEndPoint receivedIPInfo = new IPEndPoint(IPAddress.Any, 0);

    byte[] data = listener.Receive(ref receivedIPInfo);

    this.textBox1.Text += ("GOT: " +
            Encoding.ASCII.GetString(data, 0,
            data.Length) + "  FROM: " + receivedIPInfo.ToString());
}
```

3.5 Kỹ thuật Multicasting với gói tin UDP

`UDPClient` có thể dễ dàng cấu hình để broadcast tới nhiều địa chỉ IP hoặc tới gói nhận từ nhiều multicast địa chỉ IP. Từ một multicast địa chỉ IP được thao tác bằng một máy chủ, cái này duy trì một danh sách multicast subscribers. Khi một gói được gửi tới một multicast IP address, máy chủ gửi một bản sao của gói tin tới địa chỉ IP của nhiều máy khách, máy đã được tán thành.

Gửi gói Multicast

Để gửi gói UDP tới nhiều một multicast địa chỉ IP, không cần chỉ rõ hành động được yêu cầu. Đơn giản là gửi gói tin như trong ví dụ "Viết mã cho UDP client".

Nhận gói Multicast

Để nhận gói multicast, trước tiên chúng ta phải đồng ý cùng máy chủ, máy chủ được thao tác multicast địa chỉ IP. Chúng ta có đồng ý để multicast địa chỉ IP, chúng ta có thể lắng nghe gói tin từ multicast địa chỉ IP trong cách như là cho bất kỳ địa chỉ khác. Khi một ai đó gửi một gói tới multicast địa chỉ IP, máy chủ trả lại tới tất cả mọi người trên danh sách đồng ý. Để đồng ý một multicast địa chỉ IP, làm theo các bước sau:

Bước 1: Tạo một `IPAddress`, đây là một điểm để multicast địa chỉ IP.

Bước 2: Gọi `UdpClient.JoinMultiCastGroup()`

Cố gắng nhận thông tin từ multicast địa chỉ IP sẽ nhận gói tin trở lại từ multicast máy chủ. Đây là thành phần nạp chồng `JoinMultiCastGroup` hỗ trợ trên .NET Compact Framework:

- `JoinMultiCastGroup(IPAddress multicastAddr)` Kết nối một nhóm multicast ở `multicastAddr`.
- `JoinMultiCastGroup(IPAddress multicastAddr, int maxHops)` Kết nối một nhóm multicast tại `multicastAddr` nhưng chỉ nhận gói mà được tao ra bởi `maxHops`.

Ví dụ:

```
IPAddress l_multicastAddress = new IPAddress("172.68.0.22");
// Only receive multicast packets that have traveled
// for 40 or less hops
l_UDPClient.JoinMulticastGroup(l_multicastAddress, 40);
```

Để không tán thành từ một multicast địa chỉ IP, gọi `UDPClient.DropMulticastGroup()` như sau:

```
l_UDPClient.DropMulticastGroup(l_multicastAddress);
```

3.6 Truyền thông với máy chủ ở xa thông qua giao thức HTTP

Chúng ta hãy thảo luận làm thế nào làm việc với socket để truyền dữ liệu giữa máy khách và máy chủ bằng cách sử dụng gói TCP hoặc UDP. Trong mỗi trường hợp chúng ta đã đưa ra giao thức truyền thông. Ví dụ, ứng dụng quản lý chat được sử dụng một giao thức, đối tượng

ChatPacket được tạo ra thành các byte và gửi thông qua kết nối mạng. Trong ví dụ Remote Hello và UDPHello gửi một chuỗi qua lại.

Có rất nhiều máy chủ trên Internet, các máy chủ này có rất nhiều giao thức truyền thông, HTTP, các giao thức này được sử dụng trên WWW. Khi sử dụng giao thức HTTP, có rất nhiều qui tắc để làm thế nào máy khách liên lạc với máy chủ và làm thế nào để máy khách có thể đòi hỏi bất kỳ lúc nào. Dữ liệu mà máy chủ HTTP trả về cho đến khi một thiết lập gói tin TCP, nhưng sự can thiệp thông qua tất cả thông tin liên kết giao thức là một công việc hết sức buồn tẻ. Một giao dịch cùng với máy chủ HTTP có cấu trúc như sau:

Bước 1: Máy khách kết nối với máy chủ HTTP.

Bước 2: Máy chủ HTTP trả lời.

Bước 3: Máy khách yêu cầu dữ liệu bằng cách sử dụng GET hoặc yêu cầu vị trí dữ liệu bằng cách sử dụng lệnh POST.

Bước 4: Máy chủ trả về thông tin yêu cầu và dễ dàng đưa ra mã lỗi nếu yêu cầu của máy khách không thể thoả mãn. Ví dụ, mã lỗi phổ biến là 404 được trả về nếu máy khách cố gắng GET một file không tồn tại.

Bước 5: Bước 4 có số lần lặp tuỳ ý.

Bước 6: Máy khách đóng kết nối.

Mỗi lần máy khách tạo yêu cầu hoặc máy chủ trả lời, một kết nối socket mới kết nối với máy chủ được tạo. Lớp `HttpWebRequest` được tổ chức tất cả quá trình xử lý phức tạp cùng với quá trình tác động đến máy chủ HTTP. `HttpWebRequest` có thể thao tác những thao tác sau:

- Khởi tạo một kết nối với máy chủ HTTP
- Nhận kết quả trả về từ máy chủ HTTP
- Trả về một dòng lưu trữ dữ liệu được máy chủ HTTP gửi trả về như là kết quả chúng ta yêu cầu.

Sử dụng `HttpWebRequest`

Để sử dụng lớp `HttpWebRequest` để download thông tin từ máy chủ HTTP, làm theo các bước sau:

Bước 1: Tạo một thể hiện của lớp `Uri` để chỉ địa chỉ (URL) của máy chủ

Bước 2: Cài đặt một `HttpWebRequest` bằng cách sử dụng `Uri` của bước 1.

Bước 3: Yêu cầu `HttpWebRequest` trả về kết quả từ Web Server trong mẫu của lớp `Stream`.

Bước 4: Dùng nội dung của `Stream`.

Đoạn mã ví dụ về `HttpWebRequest`

Lớp `HttpWebRequest` làm giảm công việc phức tạp khi giao tiếp với máy chủ thông qua HTTP trong bốn bước trên.

```
Uri l_Uri = new Uri("http://www.myserver.com");

HttpWebRequest l_WebReq = (HttpWebRequest)WebRequest.Create(l_Uri);

HttpWebResponse l_WebResponse =(HttpWebResponse)l_WebReq.GetResponse();

Stream l_responseStream = l_WebResponse.GetResponseStream();

StreamReader l_SReader = new StreamReader(l_responseStream);

// Do something with l_SReader. For example, if you downloaded a
// Web page, you could
// extract the HTML code that came in the response and paint it on
// the screen.
```

3.7 Truyền thông với máy chủ ở xa thông qua giao thức HTTPS

Giao thức HTTPS cho phép giải quyết đảm bảo xuất hiện tại Web sites. .NET Compact Framework bổ sung thêm `HttpWebRequest` có khả năng truy cập máy chủ bằng giao thức HTTPS.

3.8 Truyền thông qua thiết bị cổng IrDA

Rất nhiều Pocket PC và các thiết bị Windows CE khác có sẵn cổng hồng ngoại (IrDA). .NET Compact Framework bao gồm các lớp để lập trình dựa vào cổng IrDA.

Truyền thông IrDA giữa hai máy tính, một máy khách (client) và một máy chủ (server). Máy chủ thường kết nối tới máy khách trong vùng của cổng hồng ngoại. Kết nối máy chủ thường được xác định bởi tên máy chủ và ID thiết bị.

Máy khách có thể trong vùng của rất nhiều máy đề nghị kết nối IrDA. Mỗi thiết bị trong vùng có thể giao tiếp có một ID và tên duy nhất. Liệt kê các máy khách thông qua các kết nối sẵn sàng, chọn một, và giao giao tiếp cùng với yêu cầu của máy tính ở xa. Giao tiếp qua cổng hồng ngoại tìm thấy cùng với thông qua các sự kiện:

Bước 1: Một thiết bị cung cấp một hoặc nhiều dịch vụ tới các máy khác trong vùng cổng IrDA của nó. Thiết bị được xác định thông qua tên và ID của thiết bị. Địch vụ cung cấp được xác định thông qua tên.

Bước 2: Một thiết bị khách muốn mở một danh sách liệt kết nối thông qua tất cả các thiết bị trong vùng của thiết bị khách.

Bước 3: Một lựa chọn máy khách của các thiết bị sẵn sàng và kết nối tới một một dịch vụ của được cung cấp bởi thiết bị đã chọn.

Sử dụng `IrDAClient` để truy cập cổng IrDA

Đối tượng trung tâm cho kết nối IrDA trên .NET Compact Framework là `IrDAClient`. Cùng với sợ giúp đỡ của nhiều lớp hỗ trợ được thảo luận, `IrDAClient` có thể làm việc như một máy chủ hoặc máy khác. `IrDAClient` có thể được sử dụng tìm kiếm các kết nối sẵn sàng hoặc các kết nối được cung cấp trên các thiết bị khác.

`IrDAClient` và các lớp liên quan IrDA tập trung trong thư viện có tên là `System.Net.IrDA.dll`. Chúng ta phải thêm một tham chiếu để sử dụng thư viện này trong dự án trước khi có thể sử dụng. Để thêm thư viện, vào menu Project-> Add References. Trong hộp thoại, bấm đúp chuột vào nhãn có tên `System.Net.IrDA` và bấm OK.

Một kết nối được tạo ra cùng với một nhóm ở xa, `IrDAClient` cung cấp phương thức `GetStream()`, phương thức này đưa ra một thể hiện `Stream` cùng với chương trình có thể đọc và ghi dữ liệu.

Kết nối tới cổng IrDA như một máy khách

Khi kết nối như một IrDA khách, nó bị lạm dung như là người phát triển, chúng ta biết tên của thiết bị cùng với cái mà chúng ta muốn kết nối. Chương trình phải lặp thông qua tất cả các thiết bị sẵn sàng và chọn một cái cùng với dịch vụ yêu cầu. Chúng ta làm theo các bước sau:

Bước 1: Tạo một kết nối `IrDAClient`.

Bước 2: Nhận danh sách các thiết bị sẵn sàng kết nối bằng cách gọi `IrDAClient.DiscoverDevices`. Phương thức `DiscoverDevices` trả về một mảng các đối tượng `IrDADeviceInfo`.

Bước 3: Duyệt mỗi `IrDADeviceInfo` trong mảng để tìm ra các thiết bị sẵn sàng được ứng dụng sẽ kết nối.

Bước 4: Nếu yêu cầu của thiết bị được tìm thấy, sau đó kết nối tới bằng cách gọi phương thức `IrDAClient.Connect()`. Thông qua tên của dịch vụ để kết nối tới.

Bước 5: Sử dụng `IrDAClient` để kết nối.

Ví dụ:

Đoạn mã sau nhận được từ ứng dụng ví dụ. Đoạn mã liệt kê tất cả thiết bị sẵn sàng và cố gắng kết nối tới một thiết bị được cung cấp bởi một dịch vụ có tên là IRDA_CHAT_SERVER. Nó là một kết nối có một chat server ở xa đợi một ai đó kết nối và chat.

```
m_IrDAClient = new IrDAClient();

bool l_foundAnyDevice = false;
int MAX_DEVICES = 5;

// Find out who's out there to connect with...
IrDADeviceInfo[] l_DevsAvailable =
        m_IrDAClient.DiscoverDevices(MAX_DEVICES);

// Show a MessageBox telling user every device we see out there
```

```
foreach (IrDADeviceInfo l_devInfo in l_DevsAvailable)
{
    l_foundAnyDevice = true;
    MessageBox.Show(l_devInfo.DeviceName, "Discovered IrDA device");

    // Now try to connect to the devices, hoping it offers a service
    // named "IRDA_CHAT_SERVER"

   try
   {
       // Assume that first device is offering a service that we
       // want
       IrDAEndPoint chatEndPoint = new IrDAEndPoint(
               l_DevsAvailable[0].DeviceID, "IRDA_CHAT_SERVER");
       m_IrDAClient.Connect(chatEndPoint);


       MessageBox.Show("Connected to chat server!", "Ready to chat");
       m_Connected = true;
       break;
    }
    catch (SocketException exc)   {    }
}

// m_IrdaClient can now be read from or written to.
```

Thiết lập một kết nối IrDA như là một máy chủ

Để thiết lập kết nối IrDA như là một thiết bị, làm theo các bước sau:

Bước 1: tạo một thể hiện của `IrDAListener`, thông qua tên của thiết bị trong cấu trúc.

Bước 2: Gọi phương thức `Start()` trên `IrDAListener`.

Bước 3: Gọi phương thức `IrDAListener.AcceptIrDAClient()` để nhận một thể hiện của `IrDAClient` khi một ai đó kết nối.

Bước 4: Sử dụng `IrDAClient` để giao tiếp với các thiết bị tham gia giao tiếp.

Thiết lập kết nối như một thiết bị chủ, ví dụ:

Đây là đoạn mã lệnh ví dụ lấy từ ứng dụng ví dụ IrDAChat. Nó demo làm thế nào để sử dụng một `IrDAListener` để cung cấp một kết nối gọi IRDA_CHAT_SERVER từ các thiết bị khác và đợi một ai đó kết nối.

```
IrDAListener l_IrDAListener = new IrDAListener("IRDA_CHAT_SERVER");

// Listen for anyone who wants to connect
l_IrDAListener.Start();

// And now pull the first queued connection request out as an
// IrDAClient
m_IrDAClient = l_IrDAListener.AcceptIrDAClient();

MessageBox.Show("Accepted a connection", "Ready to chat");
```

Đọc dữ liệu từ một `IrDAClient`

Một `IrDAClient` được kết nối với các thiết bị ở xa cùng tham gia giao tiếp, khả năng đọc dữ liệu đạt được theo cách như nhau dù kết nối chủ hay khách, như sau:

Bước 1: Tạo một `StreamReader` thông qua `Stream` đã liên kết cùng với `IrDAClient` trong cấu trúc `StreamReader`.

Bước 2: Đọc dữ liệu từ `StreamReader`.

Đọc dữ liệu từ IrDAClient: đoạn mã ví dụ:

```
l_StreamReader = new StreamReader(this.m_IrDAClient.GetStream(),
        System.Text.Encoding.ASCII);
// Read a line of text and paint it into a GUI
this.lbInText.Items.Add(l_StreamReader.ReadLine());
l_StreamReader.Close();
```

Ghi dữ liệu vào IrDAClient

Once an `IrDAClient` is connected to a remote party, writing data is achieved the same way whether connected as a server or as a client, as follows:

Bước 1: Tạo `StreamWriter` thông qua `Stream` liên kết cùng `IrDAClient` trong cấu trúc `StreamWriter`.

Bước 2: Ghi dữ liệu vào `StreamWriter`.

Ghi dữ liệu vào IrDAClient: đoạn mã ví dụ:

Sau đây là đoạn mã ví dụ lấy từ ứng dụng ví dụ IrDAChat. Đoạn mã viết một dòng văn bản, cái đó yêu lấy được từ giao diện người sử dụng, để luồng đạt được từ `IrDAClient`.

```
      // Grab a reference to the stream in the m_IrDAClient and send the
      // text to it.
      StreamWriter l_StreamWriter = new
StreamWriter(this.m_IrDAClient.GetStream(),
              System.Text.Encoding.ASCII);
      l_StreamWriter.WriteLine(this.txtSendText.Text);
      l_StreamWriter.Close();
```

Chương 4 ADO.NET trên .NET Compact Framework

4.1 Giới thiệu ADO.NET

ADO.NET là tên chỉ một tập hợp các lớp truy cập cơ sở dữ liệu trong giới lập trình .NET. Tập hợp các lớp trong ADO.NET rất nhiều. Tuy nhiên, thao tác dữ liệu đơn giản và kết nối dữ liệu từ xa có thể thao tác được chỉ với một vài dòng lệnh. Trong phần này chúng ta tìm hiểu cách thức thao tác với dữ liệu cục bộ trên thiết bị.

4.2 Lưu trữ dữ liệu bằng `DataSet`

`DataSet` là các lớp framework cơ bản để thao tác dữ liệu cùng với .NET Compact Framework. `DataSet` có thể coi như là bộ máy cơ sở dữ liệu quan hệ trong chính nó. Nó lưu dữ các bảng trong bộ nhớ được sắp xếp như là các bảng, dòng, và cột và cho phép người phát triển thực hiện các thao tác cơ sở dữ liệu chuẩn, như thêm và xóa dữ liệu, sắp xếp, và kiểm tra ràng buộc.

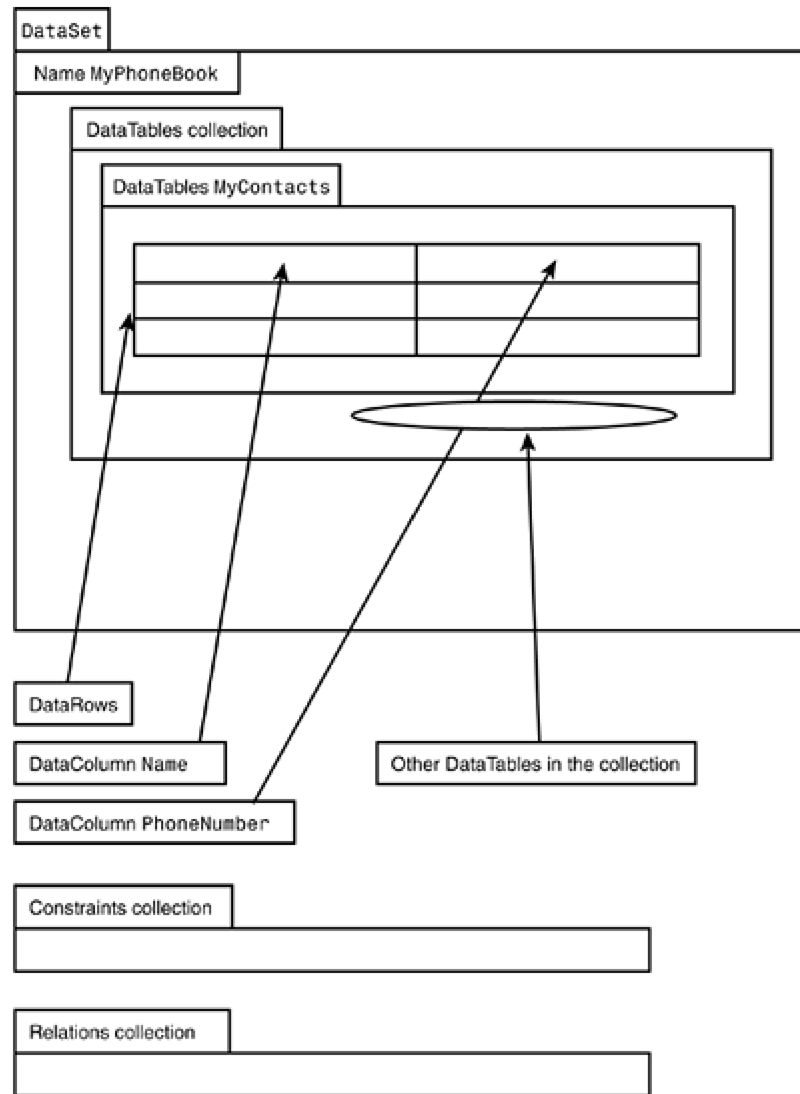
Các nhà phát triển là những người hiểu làm thế nào để làm việc có hiệu quả với `DataSet` trên .NET Compact Framework sẽ ghi hiệu quả các ứng dụng ADO.NET trên framework.

Để thao tác dữ liệu trong lập tình ADO.NET được đổ vào `DataSet` từ CSDL lớn, làm việc với các khoang dữ liệu trong `DataSet`, và ghi dữ liệu thay đổi trở lại CSDL. Trong phần này chúng ta tìm hiểu làm thế nào để đổ dữ liệu vào `DataSet` bằng cách trèn dữ liệu chương trình và thực hiện thao tác đơn giản trên dữ liệu.

4.2.1 Bên trong `DataSet`: `DataTables`, `DataRows`, và `DataColumns`

`DataSet` chứa một hoặc nhiều `DataTable`s. Mỗi `DataTable`s tương ứng với một bảng trong CSDL quan hệ. Nó có một tập hợp các `DataRow`s, và mỗi `DataRow` có một tập hợp `DataColumn`s trên thực tế lưu trữ dữ liệu. Để tạo `DataSet`, `DataTable`s, và `DataColumn`s rất đơn giản. Hình 4.1 mô hình kiến trúc tổng quan cách một `DataSet` lưu trữ dữ liệu đơn giản phonebook.

Hình 4.1. Miêu tả `DataSet` cấu trúc của phone book.



Chúng ta có thể sử dụng riêng một `DataTable` để lưu trữ dữ liệu liên quan cùng với một bảng, nhưng `DataSet` cung cấp các phương thức và thuộc tính có tiện ích thêm và thực sự tạo một CSDL quan hệ thu nhỏ trong bộ nhớ. Ví dụ, cùng với `DataSet` chúng ta có thê làm việc tất cả các cái sau:

- Thao tác với thông tin bên trong một `DataSet` như là một CSDL quan hệ nhỏ. Ví dụ, chúng ta có thể thiết lập mối quan hệ cha con, cập nhật, xóa, và tạo cột dữ liệu được tính toán từ các trường dữ liệu khác.
- Ghi và lạp nội dung của tất cả `DataTables` vào một file XML cùng với chỉ một dòng lệnh.
- Thông qua bộ máy SQL CE, bộ máy này sẽ giúp chúng ta đưa vào nó các bảng từ CSDL quan hệ được lưu trữ trên thiết bị hoặc thay thế bằng dữ liệu từ máy chủ ở xa.
- Thông qua nó SQL cung cấp được đưa vào các bảng từ máy chủ ở xa.
- Nhận một phần của `DataSet`, cái mà trả về giá trị của dịch vụ Web, hoặc thông qua `DataSet` trở lại dịch vụ Web.

Trong phần này chúng ta sẽ tìm hiểu cách thao tác dữ liệu trên `DataSet`.

4.2.2 Đưa dữ liệu vào DataSet

Để đưa dữ liệu vào một `DataSet`, chúng ta làm theo các bước sau:

Bước 1: Nhận một thao tác DataTable với DataSet mà chúng ta muốn thêm một dòng mới vào. Nếu cần thiết, tạo một DataTable mới. Tập hợp các DataTables mà DataSet quản lý có thể được sử dụng thông qua thuộc tính DataSet.Tables. Nếu chúng ta phải tạo một DataTable mới và thêm nó vào tập hợp DataSet.Table, sao đó thông thường chúng ta thêm dữ liệu vào bảng đã tồn tại, vì vậy chúng ta có thể bỏ qua các bước này:

Tạo một DataTable thông qua cấu trúc DataTable.

Tạo một DataColumns và thêm vào nó tập hợp DataTable.Columns. Mỗi DataColumn, chúng ta phải chỉ ra tên, kiểu dữ liệu của cột.

Thêm DataTable vào tập hợp DataSet.Tables bằng cách gọi phương thức .Add.

Bước 2: Tạo một DataRow. Ví dụ, tạo một dòng mới cho DataTable đầu tiên trong DataSet:

Sử dụng l_newRow = l_DataSet. Tables[0].NewRow.

Chúng ta có thể chỉ ra ra bảng thông qua tên bảng:

Ví dụ _newRow = l_DataSet.Tables["Customers"].NewRow

Bước 3: DataRow mới tự động được tạo cùng với các cột vào tương ứng với bảng đã được tạo. Trèn giá trị vào các cột của DataRow.

Bước 4: Thêm DataRow mới vào tập hợp Rows của DataTable đã được tạo:

Ví dụ _DataSet.Tables[0].Rows.Add(l_newRow);.

Bước 5: Sau khi chúng ta đã thêm vào tất cả các dòng như mong muốn, gọi phương thức DataSet.AcceptChanges để đồng ý tất cả sự thay đổi. Để hủy bỏ tất cả việc thêm mới dữ liệu chúng ta gọi phương thức DataSet.RejectChanges.

4.2.3 Xây dựng một DataSet lưu trữ một Phone Book

Để mô tả cách tạo một DataSet có khả năng lưu trữ dữ liệu quan hệ, chúng ta hãy xem xét ví dụ ứng dụng PhoneBook. Trong ứng dụng ví dụ này, chúng ta hãy xem xét khả năng của một DataSet lưu trữ phone book. DataSet lưu trữ một bảng DataTable, bảng này được thiết lập gồm hai DataColumns, cột thứ nhất lưu trữ tên và cột thứ hai lưu trữ số điện thoại. Đoạn mã sau mô tả năm bước cần thiết để thêm dữ liệu vào một DataSet, bao gồm tạo một bảng mới.

Listing 4.1 Tạo và lưu trữ một DataSet

```
DataSet     l_DataSet = new DataSet();

// Create a DataTable that holds a "Name" and a "PhoneNumber"
DataTable  l_newTable = new DataTable("Phone Contacts");
        l_newTable.Columns.Add(new DataColumn("Name",
        typeof(System.String)));
```

```
l_newTable.Columns.Add(new DataColumn("PhoneNumber",
        typeof(System.String)));

// Add the DataTable to the DataSet's table collection
l_DataSet.Tables.Add(l_newTable);

// Now put a few names in...
// GEORGE WASHINGTON
DataRow l_newRow = l_DataSet.Tables[0].NewRow();
l_newRow[0] = "George Washington";
l_newRow[1] = "555 340-1776";
l_DataSet.Tables[0].Rows.Add(l_newRow);

// BEN FRANKLIN
l_newRow = l_DataSet.Tables[0].NewRow();
l_newRow["Name"] = "Ben Franklin";
l_newRow["PhoneNumber"] = "555 336-3211";

l_DataSet.Tables[0].Rows.Add(l_newRow);

// Commit the changes
l_DataSet.AcceptChanges();
```

4.2.4 Trích dữ liệu từ một `DataSet`

Trích dữ liệu từ một `DataSet` như là truy nhập `DataTable` trong tập hợp `DataSet.Tables` và tìm kiếm dòng mong muốn trong bảng. Mỗi dòng có một chỉ số, tạo cho nó dễ dàng truy cập cột mong muốn. Chỉ số đầu tiên là 0, như trong ví dụ:

`l_DataSet.Tables[0].Rows[0][0]` Truy nhập cột đầu tiền trong dòng đầu tiên của `DataTable` đầu tiên.

`l_DataSet.Tables[0].Rows[0][9]` truy nhập cột thứ 10 trong dòng đầu tiên của `DataTable` đầu tiền.

`l_DataSet.Tables[0].Rows[29][9]` Truy nhập cột thứ 10 trong dòng 30 của `DataTable` đầu tiền.

Trích dữ liệu PhoneBook từ một `DataSet`

Sau đây là đoạn mã trong ứng dụng ví dụ PhoneBook. Nó tìm kiếm thông qua tất cả các dòng trong `DataTable` đầu tiên trong một `DataSet` và đưa giá trị cột thứ 0 và thứ 1 vào một ListBox:

```
for (int i = 0; i < phonebookEntriesDataSet.Tables[0].Rows.Count; i++)
{
   this.listBox1.Items.Add(
        phonebookEntriesDataSet.Tables[0].Rows[i][0] + "   " +
        phonebookEntriesDataSet.Tables[0].Rows[i][1]);
}
```

4.2.5 Thay đổi dữ liệu trong một `DataSet`

Để thay đổi dữ liệu trong `DataSet`, truy cập vào `DataColumn` mà chúng ta muốn thay đổi và thiết lập giá trị mới. Khi tất cả thay đổi đã kết thúc, gọi `AcceptChanges` để xác nhận sự thay đổi.

Ví dụ, đoạn mã sau thiết lập cột thứ 2 trong dòng đầu tiên của bảng đầu tiên trong tập hợp `DataSet` thành một số ngâu nghiện được cung cấp bởi `randomGenerator`, `randomGenerator` được cung cấp trong lớp `Random`.

```
// Column 1 is the phone number.
//                                |
//                                V
m_phonebookDS.Tables[0].Rows[0][1] = randomGenerator.Next().ToString();
```

Thay đổi bằng cách sử dụng chỉ số tên, cách này sẽ chậm hơn trong .NET Compact Framework khi lượng lớn dữ liệu phức tạp:

```
m_phonebookDS.Tables["Phone Contacts"].Rows[0]["PhoneNumber"] =
  l_randomGenerator.Next().ToString();
```

4.3 Ràng buộc dữ liệu

`DataSet` cho phép chúng ta chỉ ra qui tắc riêng biệt, mà dữ liệu trong tập hợp `DataSet.Tables` phải theo. Lớp cơ sở `Constraint` chỉ rõ qui tắc mà dữ liệu trong `DataTable` phải theo.

`ForeignKeyConstraint` thường được sử dụng để tạo sức mạnh cho hành vi khi thay đổi hoặc xóa cột khóa chính trong một bảng. Bởi vì `ForeignKeyConstraint` được mong đợi để sử dụng trong mô hình quan hệ cha con giữa các bảng.

4.3.1 Thêm ràng buộc vào một `DataSet`

Mỗi `DataTable` lưu trữ trong tập hợp `DataSet.Tables` lưu trữ `ConstraintCollection` trong thuộc tính `Constraints`. Vsi dụ, để truy cập `ConstraintCollection` trong bảng đầu tiên của một `DataSet`, sử dụng như sau:

```
m_phonebookDS.Tables[0].Constraints
```

Các bước để tạo và khởi tạo ràng buộc:

Bước 1: Thêm ràng buộc vào tập hợp `Constraints` của bảng thíc hợp.

Bước 2: Thiết lập cờ `DataSet.EnforceConstraints` thành true để bật yêu cầu ràng buộc. Khi chúng ta thiết lập cờ thành true, mỗi ràng buộc trong mỗi tập hợp `DataTable.Constraints` được chọn, và đưa ra một ngoại lệ nếu kiểm tra bị lỗi.

4.3.2 Thêm một `UniqueConstraint`

Để thêm một `UniqueConstraint` vào một `DataSet`, làm theo các bước sau:

Bước 1: Tạo một `UniqueConstraint` bằng cách sử dụng một trong bốn khởi tạo trên .NET Compact Framework:

`UniqueConstraint(String name, DataColumn col)` Creates a `UniqueConstraint` with specified name that enforces uniqueness on a single `DataColumn`.

`UniqueConstraint(DataColumn col)` Creates a `UniqueConstraint` that enforces uniqueness on a single `DataColumn`.

`UniqueConstraint(String name, DataColumn[] cols)` Creates a `UniqueConstraint` that enforces uniqueness for multiple columns in a row. The columns are specified by passing them as an array.

`UniqueConstraint(DataColumn[] cols)` Same as above except the `UniqueConstraint` is nameless.

`UniqueConstraint(String name, string[] colNames, bool isPrimaryKey)` This fifth public constructor is useful only to the Smart Device Extensions environment.

Bước 2: Thêm `UniqueConstraint` vào tập hợp `Constraints` của `DataTable` mong muốn.

Bước 3: Thiết lập `DataSet.EnforceConstraints` thành true để bật sự ràng buộc..

Ví dụ:

```
    // Add a UniqueConstraint to the phone number column
    // Note: Using indexing by the string "PhoneNumber" is slower
    UniqueConstraint l_UniqueConstraint = new
UniqueConstraint(l_DataSet.Tables[0].
              Columns["PhoneNumber"]);
    l_DataSet.Tables[0].Constraints.Add(l_UniqueConstraint);
```

4.3.3 Ngăn ngừa giá trị NULL trong DataColumn

Thuộc tính `DataColumn.AllowDBNull` rất hữu ích để không cho phép một `DataColumn` có giá trị `DBNull`. Nếu chúng ta tạo một `DataRow` mới và không đưa một giá trị vào một cột, nó nhận giá trị mặc định là `DBNull`.

Ví dụ:

```
l_newTable.Columns["Name"].AllowDBNull = false;
```

Nếu DataColumn có `AllowDBNull là false` được thiết lập thành `DBNull`, ngoại lệ `System.Data.NoNullAllowed` được đưa ra khi một dòng mới được thêm vào DataTable trong `DataSet`. Ví dụ:

```
DataRow l_newRow = m_phonebookDS.Tables[0].NewRow();
l_newRow[0] = "Violator"
l_newRow[1] = "5555587";

// This is going to throw an exception because the "Name"
// DataColumn was never set, so it is DBNull, and that is
// not allowed for the DataColumn
m_phonebookDS.Tables[0].Rows.Add(l_newRow);]
```

4.4 Thiết lập trường tự động tăng giá trị

Khi một dòng được thêm vào `DataTable`, dòng rỗng được tạo bằng cách gọi `DataTable.NewRow`. Một `DataTable` biết giản đồ cho một dòng, dòng này phải được tạo và thể

hiện dòng mới tương ứng với giản đồ. Có nghĩa là dòng mới lưu trữ phải có DataColumns cùng với kiểu dữ liệu đúng với cái mà chúng ta đã thiết lập.

Thuộc tính DataColumn.AutoIncrement có thể được thiết lập trong DataTable là một DataColumn có giá trị tự động tăng khi một dòng mới được tạo. Nó rất hữu dụng khi làm trường khóa.

Có ba thuộc tính quan trong trong DataColumn liên quan đến trường tự động tăng giá trị:

DataColumn.AutoIncrement Thiết lập giá trị true cho DataColumn tự động tăng.

DataColumn.AutoIncrementSeed Giá trị bắt đầu cho giá tự động tăng.

DataColumn.AutoIncrementStep Giá trị của bước nhảy cho mỗi giá trị mới.

Nếu DataColumn là một cột tính tóan, sau đó cố gắng thiết lập như là một cột tự động tăng sẽ nguyên nhân một ArgumentException.

Nếu kiểu dữ liệu của DataColumn không phải là Int16, Int32, hoặc Int64, sau đó nó bị ép kiểu thành Int32. Nó có thể là nguyên nhân mất dữ liệu, nếu DataColumn là kiểu số thực. Nếu DataColumn là kiểu chuỗi, thiết lập cột đó tự động tăng giá trị sẽ ép kiểu dữ liệu của cột này thành kiểu integer.

Ví dụ tạo một trường có giá trị tự động tăng:

Thiết lập một trường có giá trị tự động tăng từ 10, bước nhảy có giá trị là 5.

```
l_newTable.Columns["ContactID"].AutoIncrement = true;
l_newTable.Columns["ContactID"].AutoIncrementSeed = 10;
l_newTable.Columns["ContactID"].AutoIncrementStep = 5;
```

4.5 Mô hình dữ liệu quan hệ với DataSet

Chúng ta hãy tìm hiểu DataSet lưu trữ DataTable, truy nhập dữ liệu, và yêu cầu theo mẫu ràng buộc trên dữ liệu. Trong phần này chúng ta xây dựng kiến thức và học các thao tác nền tảng chung nhất về CSDL quan hệ cùng với dữ liệu bên trong DataSet.

Xuất phát từ giá trị DataColumn cùng với biểu thức và trường tính toán

Giá trị của DataColumn có thể được tính toán dựa trên giá trị của DataColumn khác trong cùng một DataRow. Để làm điều này, sử dụng thuộc tính DataColumn.Expression để mô tả giá trị tính toán của DataColumn. Thuộc tính Expression là một giá trị chuỗi được mô tả sự tính toán xuất phát từ giá trị cho DataColumn.

Cú pháp biểu thức rất nhiều và hỗ trợ rất nhiều phép tính toán học và chuỗi. Bảng 4.1 đưa đến tất cả các phép toán được .NET Compact Framework hỗ trợ.

Bảng 4.1. Các phép toán Framework hỗ trợ để tính toán

Phép toán	Chức năng

Bảng 4.1. Các phép toán Framework hỗ trợ để tính toán

Phép toán	Chức năng
`Sum`	Tính tổng các đối số
`Avg`	Tính trung bình các đối số
`Min`	Lựa chọn giá trị nhỏ nhất của các đối số
`Max`	Lựa chọn giá trị nhỏ lớn của các đối số
`+, -, *, /`	Cộng, trừ, nhân, chia
`%`	Phép chia lấy phần dư
`+`	Ghép chuỗi

Ví dụ:

```
      l_newTable.Columns["FullName"].Expression = "FirstName + ' ' +
LastName";
      l_newTable.Columns["TotalPrice"].Expression = "MSRP - Discount";
      l_newTable.Columns["FinalGrade"].Expression = "Avg(Exam1, Exam2,
Exam3)";
```

Biểu thức quan hệ cha con trong `DataSet`

Thực chất thành phần của CSDL quan hệ là các bảng với các dòng có khả năng tạo quan hệ cha con, hoặc một quan hệ, giữa hai bảng. Một quan hệ giưa hai bảng được tạo bằng liên kết giưa hai bảng bằng một hoặc nhiều cột dữ liệu gọi là khóa chính. Trong bảng cha, khóa chính xác định mỗi dòng là duy nhất trong bảng. Các dòng trong bảng con có một trường gọi là khóa ngoại, trường này không phải là duy nhất trong bảng con.

Ví dụ bảng cha `MainContactTable`, và bảng con `CholesterolTable`.

Bảng 4.2. MainContactTable

Tên trường	Kiểu dữ liệu
`CustID`	Integer, Khóa chính
`FirstName`	String
`LastName`	String

Bảng 4.3. CholesterolTable

Tên trường	Kiểu dữ liệu
`CustID`	Integer, Khóa chính
`Reading1`	Decimal
`Reading2`	Decimal
`Reading3`	Decimal
`Average`	Decimal

Trong bảng `CholesterolTable`, `CustID` tham chiếu đến một bản ghi duy nhất trong bảng `MainContactTable`. Bảng 4.4 và 4.5 cho thấy quan hệ cha con khi lưu trữ.

Bảng 4.4. `MainContactTable`

CustID	FirstName	LastName
001	George	Washington
002	Ben	Franklin
003	Alexander	Hamilton

Bảng 4.5. `CholesterolTable`

CustID	Reading1	Reading2	Reading3	Average
001	87	78	66	77.0
001	99	54	89	80.667
002	90	88	55	77.667

Trong ví dụ bảng cha con ví dụ trên, bản ghi trong bảng `CholesterolTable` tương ứng với George Washington và một bản ghi tương ứng với Ben Franklin. Vấn đề gì sẽ xây ra nếu bản ghi George Washington bị xóa trong bảng `MainContactTable`? Hệ thống sẽ bị xóa tất cả bản ghi tương ứng trong bảng `CholesterolTable`, hoặc CSDL sẽ ở trạng thái lỗi.

.NET Compact Framework cung cấp hai lớp có thể làm việc đó tự động: `DataRelation` và `ForeignKeyConstraint`.

Tạo một `DataRelation` để thể hiện quan hệ cha con

Khi thiết lập một `DataRelation` giữa hai bảng, chúng ta chỉ rõ `DataColumn` như là khóa chính và khóa ngoại. Sau khi `DataRelation` được tạo, nó sẽ đảm bảo rằng dữ liệu quan hệ của `DataSet` như là được mô tả bởi `DataRelation`. Ví dụ, nếu chúng ta xóa bản ghi đầu tiên trong bảng `MainContactTable`, `DataRelation` sẽ tự động xóa tất cả các dòng con trong bảng `CholesterolTable`.

Để thiết lập `DataRelation` giữa hai bảng trong một `DataSet`, trước tiên tạo `DataRelation` bằng cách sử dụng hàm khởi tạo thông qua `DataColumn`s bao gồm khóa chính và khóa ngoại. Các hàm khởi tạo .NET Compact Framework như sau:

- `DataRelation(String relName, DataColumn parent, DataColumn child)` Tạo một `DataRelation` giữa `DataColumn`s cha và con.
- `DataRelation(String relName, DataColumn[] parent, DataColumn[] child)` Tạo `DataRelation` giữa hai bảng sử dụng nhiều trường cho mỗi bảng để quan hệ.
- `DataRelation(String relName, DataColumn parent, DataColumn child, bool createConstraints)` Tạo một `DataRelation` giữa `DataColumn`s cha và con.
- `DataRelation(string relName, DataColumn[] parent, DataColumn[] child, bool createConstraints)` Tạo `DataRelation` giữa hai bảng bằng cách sử dụng nhiều cột trong mỗi bảng cho liên kết.

- `DataRelation(string relName, string parentTableName, string childTableName, string[] parentColNames, string[] childColNames, bool isNested)` là một khởi tạo đã sử dụng môi trường Smart Device Extensions.

Viết mã lệnh để tạo DataRelation

```
DataRelation l_newRelation = new DataRelation(
"MainContactToCholesterolRelation",
l_DataSet.Tables["PhoneContactsMainTable"].Columns["ContactID"],
l_DataSet.Tables["Cholesterol"].Columns["ContactID"]);
l_DataSet.Relations.Add(l_newRelation);
```

4.6 Gắn dữ liệu với các điều khiển

Gắn dữ liệu với `DataGrid`

Khi `DataSet` được giới hạn vào `DataGrid`, nội dung của `DataSet` sẽ tự động xuất hiện trên `DataGrid`. Để gắn `DataSet` với `DataGrid`, chúng ta làm theo các bước sau:

Bước 1: Tạo một `DataView`.

Bước 2: Kéo một `DataGrid` từ hộp thoại công cụ.

Bước 3: Thiết lập thuộc tính `DataGrid.DataSource` với `DataView` chúng ta đã tạo ở bước 1.

Ví dụ sau đây cho thấy cách gắn một `DataGrid` với `DataView`.

```
// Assuming that m_DataSet was already set up...
m_sortAgeDataView = new DataView(m_DataSet.Tables[0]);
m_sortAgeDataView.Sort = "Age DESC, Name DESC";

// Bind the DataGrid to our DataView and it will
// automatically paint itself!

dataGrid1.DataSource = m_sortAgeDataView;
```

Chương 5 Lập trình với Microsoft SQL Server CE

5.1 Tìm hiểu các tính chất hỗ trợ bởi Microsoft SQL Server 2000 Windows CE Edition

Ngôn ngữ truy vấn có cấu trúc (SQL) Server 2000 Windows CE Edition (SQL Server CE) rất nhỏ so với bộ máy CSDL Microsoft's SQL Server 2000. Mặc dù kích cỡ của nó như vậy, nhưng SQL Server CE cung cấp đủ để lưu trữ dữ liệu và các chức năng.

SQL Server CE hỗ trợ CSDL có dung lượng lớn nhất đến 2GB. SQL Server CE hỗ trợ tập con các ngôn ngữ định nghĩa dữ liệu và ngôn ngữ thao tác dữ liệu. Có hỗ trợ nhiều cột chỉ số, khóa chính, ràng buộc.

Khi phát triển ứng dụng SQL Server CE, chúng ta cần phải thêm hai assembly references để dự án của chúng ta làm việc như đoạn mã. SQL Server CE quản lý sự tồn tại `System.Data.SqlServerCe`. Chúng ta sẽ cần thêm một tham chiếu `System.Data.Common`. Như trong đoạn mã sau:

```
using System.Data;
using System.Data.Common;
using System.Data.SqlServerCe;
```

5.2 Tạo CSDL Microsoft SQL Server CE

Có hai lựa chọn để tạo CSDL SQL Server CE. Một là sử dụng SQL Server CE Query Analyzer để dùng đồ họa tạo và thiết kế CSDL SQL Server CE. Để học nhiều hơn về Query Analyzer, xem Microsoft SQL Server CE Books Online.

Chúng ta có thể tạo một CSDL SQL Server CE bằng cách lập trình sử dụng lớp SQL Server CE Data Provider định nghĩa trong không gian tên `System.Data.SqlServerCE`. Khi tạo một CSDL bằng cách lập trình, chúng ta chỉ cần tác động đến lớp SQL Server CE Data Provider, `System.Data.SqlServerCe.SqlCeEngine`. Lớp `SqlCeEngine` cung cấp khả năng lập trình truy nhập SQL Server CE. `SqlCeEngine` cung cấp hai chức năng chính: khả năng tạo một CSDL mới và khả năng compact một CSDL đã có.

Để tạo một CSDL SQL Server CE bằng cách lập trình rất đơn giản. Chúng ta làm theo ba bước sau:

Bước 1: Trước tiên chúng ta đảm bảo răng chưa tồn tại fiel CSDL (.sdf) trước khi tạo CSDL. Nếu tồn tại, hãy xóa khi bạn tạo CSDL mới.

Bước 2: Thể hiện lớp `SqlCeEngine` phải được cài đạt và khởi tạo cùng với chuỗi kết nối.

Bước 3: Gọi phương thức `CreateDataBase` trên `SqlCeEngine`.

Listing 5.2 Tạo một CSDL SQL Server CE

```
public void CreateNewDatabase() {
  if(File.Exists("tempdb.sdf")
    File.Delete("tempdb.sdf");
```

```
  string connStr = "Data Source = tempdb.sdf; Password = testing123"

  using(SqlCeEngine engine = new SqlCeEngine(connStr)) {
    engine.CreateDatabase();
  }
}
```

5.3 Thêm cấu trúc vào một CSDL Microsoft SQL Server CE

Sau khi tạo một CSDL SQL Server CE, bước tiếp theo thêm các bảng vào CSDL. Chúng ta có thể dùng đồ họa bằng cách sử dụng SQL Server CE Query Analyzer hoặc bằng cách lập trình sử dụng lớp SQL Server CE Data Provider.

Để lập trình tạo bảng CSDL, chúng ta sẽ cần kết nối với CSDL bằng cách sử dụng lớp `SqlCeConnection` và đưa ra các câu lệnh DDL bằng cách sử dụng lớp `SqlCeCommand`.

SQL Server CE hỗ trợ một tập con của DDL. Bảng 5.2 mô tả các câu lệnh DDL hỗ trợ.

Bảng 5.2. Các câu lện DDL hỗ trợ bởi SQL Server CE

Câu lệnh DDL	Chức năng
`CREATE DATABASE`	Tạo mới CSDL và file được sử dụng lưu trữ CSDL.
`CREATE TABLE`	Tạo bảng mới. Khóa chính, và khóa ngoại, và giá trị mặc định được chỉ ra trong câu lệnh này.
`ALTER TABLE`	Thay đổi định nghĩa bảng bằng cách thay đổi, thêm, hoặc xóa cột và ràng buộc.
`CREATE INDEX`	Tạo một chỉ số trên bảng nhất định.
`DROP INDEX`	Loại bỏ một hoặc nhiều chỉ số từ CSDL hiện tại.
`DROP TABLE`	Loại bỏ một bảng và tất cả dữ liệu, chỉ số, và ràng buộc trong bảng.

Các kiểu dữ liệu SQL Server CE hỗ trợ.

Bảng 5.3. Các kiểu dữ liệu SQL Server CE hỗ trợ

Kiểu dữ liệu	Mô tả
`Bigint`	Integer (whole number) data from -2^{63} (–9,223,372,036,854,775,808) through $2^{63} - 1$ (9,223,372,036,854,775,807).
`Integer`	Integer (whole number) data from -2^{31} (–2,147,483,648) through $2^{31} - 1$ (2,147,483,647).
`Smallint`	Integer data from –32,768 to 32,767.
`Tinyint`	Integer data from 0 to 255.
`Bit`	Integer data with either a 1 or 0 value.
`numeric (p, s)`	Fixed-precision and scale-numeric data from $-10^{38} + 1$ through $10^{38} - 1$. p specifies precision and can vary between 1 and 38. s specifies scale and can vary between 0 and p.

Bảng 5.3. Các kiểu dữ liệu SQL Server CE hỗ trợ

Kiểu dữ liệu	Mô tả
Money	Monetary data values from -2^{63}/10,000 through $(2^{63}-1)$/10,000 (–922,337,203,685,477.5808 through 922,337,203,685,477.5807 units).
Float	Floating-point number data from –1.79E+308 through 1.79E+308.
Real	Floating precision number data from –3.40E+38 through 3.40E+38.
Datetime	Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of one three-hundredth second, or 3.33 milliseconds. Values are rounded to increments of .000, .003, or .007 milliseconds.
nchar(n)	Fixed-length Unicode data with a maximum length of 255 characters. Default length = 1.
nvarchar(n)	Variable-length Unicode data with a length of 1 to 255 characters. Default length = 1.
ntext	Variable-length Unicode data with a maximum length of $(2^{30}-2)/2$ (536,870,911) characters.
binary(n)	Fixed-length binary data with a maximum length of 510 bytes. Default length = 1.
varbinary(n)	Variable-length binary data with a maximum length of 510 bytes. Default length = 1.
Image	Variable-length binary data with a maximum length of $2^{30}-1$ (1,073,741,823) bytes.
uniqueidentifier	A globally unique identifier (GUID).
IDENTITY [(s, i)]	This is a property of a data column, not a distinct data type. Only data columns of the integer data types can be used for identity columns. A table can have only one identity column. A seed and increment can be specified, and the column cannot be updated. s (seed) = starting value i (increment) = increment value
ROWGUIDCOL	This is a property of a data column, not a distinct data type. It is a column in a table that is defined by using the `uniqueidentifier` data type.

Bây giờ chúng ta học cách tạo cấu trúc một CSDL SQL Server. Chúng ta tạo CSDL bao gồm hai bảng: bảng `Package` và bảng `TrackingEntry`. Bảng 5.4 và 5.5 mô tả các cột và kiểu dữ liệu tương ứng.

Bảng 5.4. Cấu trúc bảng `Package`

Tên cột	Kiểu	Kích cỡ
ID	Int	IDENTITY(1,1) PRIMARY KEY
Code	Nvarchar	12
DestinationID	Nvarchar	12

Bảng 5.5. Cấu trúc của bảng `TrackingEntry`

Tên cột	Kiểu	Kích cỡ

Bảng 5.5. Cấu trúc của bảng `TrackingEntry`

Tên cột	Kiểu	Kích cỡ
ID	Int	IDENTITY(1,1) PRIMARY KEY
PackageID	Int	FOREIGN KEY
LocationID	Nvarchar	12
ArrivalTime	Datetime	
DepartureTime	Datetime	

Listing 5.3 Tạo bảng `Package` và `TrackingEntry`

```
public static void CreateTrackingDatabase() {
  string connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connstr)) {
    conn.Open();

    // Create an the package table
    string ddlPackage =
      "CREATE TABLE Package( " +
      "ID int not null identity(1,1) PRIMARY KEY, " +
      "Code nvarchar(12) not null, " +
      "DestinationID nvarchar(12) not null)";
      RunDDLCommand(conn, ddlPackage);

      // Create the tracking entry table
      string ddlTrackingEntry =
        "CREATE TABLE TrackingEntry( " +
        "ID int not null identity(1,1), " +
        "PackageID int not null, " +
        "LocationID nvarchar(12) not null, " +
        "ArrivalTime datetime not null, " +
        "DepartureTime datetime null, " +
        "FOREIGN KEY (PackageID) REFERENCES Package(ID) )";
      RunDDLCommand(conn, ddlTrackingEntry);

      // Create an index on the tracking entry table
      string ddlArrivalTimeNdx =
        "CREATE INDEX ArrivalTime ON TrackingEntry(ArrivalTime )";
      RunDDLCommand(conn, ddlArrivalTimeNdx );
  }
}
```

Phương thức bắt đầu để tạo một kết nối tới CSDL SQL Server là đối tượng `SqlCeConnection`. Đối tượng thể hiện được tạo bằng cách sử dụng chuỗi kết nối truy cập vào CSDL. Tiếp theo kết nối tới CSDL được mở bằng cách gọi phương thức: `SqlCeConnection.Open`. Chúng ta tạo bảng `Package`. Sử dụng chuỗi câu lệnh SQL để tạo bảng. Tạo bảng `TrackingEntry`. Bảng này chứa khóa ngoại ràng buộc trên cột `PackageID`. Giá trị trèn vào cột `PackageID` phải tồn tại trong cột `ID` của bảng `Package`.

Phương thức `RunDDLCommand` tạo các yếu tố khác nhau của CSDL.

Listing 5.4 Phương thức thực thi `RunDDLCommand`

```
public static void
RunDDLCommand(SqlCeConnection conn, string ddlCmdStr) {
```

```
  SqlCeCommand cmdDDL = null;

  try {
    cmdDDL = new SqlCeCommand(ddlCmdStr, conn);
    cmdDDL.CommandType = CommandType.Text;
    cmdDDL.ExecuteNonQuery();
  } catch(SqlCeException scee) {
    for(int curExNdx = 0; curExNdx < scee.Errors.Count; ++curExNdx) {
      MessageBox.Show("Error:"+scee.Errors[curExNdx].ToString()+"\n");
    }
  } finally {
    if( cmdDDL != null )
      cmdDDL.Dispose();
  }
}
```

Table 5.6. The `CommandType` Enumeration Values

Tên	Mô tả
`StoreProcedure`	Tên của thủ stored procedure. SQL Server CE không hỗ trợ stored procedures.
`Text`	Một câu lệnh SQL text.
`TableDirect`	Khi thuộc tính `CommandType` được thiết lập `TableDirect`, thuộc tính sẽ được thiết lập tên của bảng hoặc bảng được truy cập. Tất cả dòng và cột của bảng hoặc bảng sẽ trả về khi chúng ta gọi phương thức `Execute`.

5.4 Lưu trữ (Populating) CSDL Microsoft SQL Server CE

Một CSDL SQL Server CE có thể được quản lý bằng các câu lệnh quản lý dữ liệu SQL. SQL Server CE 2.0 hỗ trợ tập con các câu lệnh quản lý dữ liệu của SQL Server. Các câu lệnh hỗ trợ được liệt kê trong bảng 5.7.

Bảng 5.7. Câu lệnh DML hỗ trợ bởi SQL Server CE

Câu lệnh	Chức năng
`INSERT`	Thêm dòng mới vào bảng
`UPDATE`	Thay đổi dữ liễu đã tồn tại trong bảng.
`DELETE`	Xóa dòng trong bảng
`SELECT`	Lấy thông tin từ CSDL và cho phép lựa chọn một hoặc nhiều dòng hoặc cột từ một hoặc nhiều bảng. Câu lệnh `SELECT` hỗ trợ kết nối trong và kết nối ngoài, và `Order By`, `Group By`, và mệnh đề `Having`.

SQL Server CE Query Analyzer có thể sử dụng các câu lệnh DML. Lớp `SqlCeCommand` có thể sử dụng thực thi trong lập trình thông qua SQL Server CE Data Provider.

Để quản lý CSDL SQL Sever CE, chúng ta có thể chạy các câu lệnh `INSERT`. Các bước như sau:

Bước 1: Mở một kết nối CSDL SQL Server CE sử dụng thể hiện của lớp `SqlCeConnection`.

Bước 2: Tạo đối tượng `SqlCeCommand`, và đưa chuỗi câu lệnh `INSERT`.

Bước 3: Thiết lập kiểu câu lệnh, thực thi câu lênh bằng cách sử dụng phương thức ExecuteNonQuery.

Listing 5.5 Mô tả cách trèn dữ liệu vào bảng Package.

```
public static void
InsertNewPackage(string pckgCode, string destID) {
  String connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connStr)) {
    conn.Open();

    string dmlInsertPackage =
      "INSERT INTO Package(Code, DestinationID) " +
      "VALUES ('" + pckgCode + "', '" + destID + "')";
    SqlCeCommand cmdInsertPackage =
      new SqlCeCommand(conn, dmlInsertPackage);

    try {
      cmdInsertPackage = new SqlCeCommand(conn , dmlInsertPackage);
      cmdInsertPackage.CommandType = CommandType.Text;
      cmdInsertPackage.ExecuteNonQuery();
    } catch(SqlCeException scee) {
      for(int curNdx=0; curNdx<scee.Errors.Count; ++curNdx) {
        MessageBox.Show("Error:"+scee.Errors[curNdx].ToString()+"\n");
      }
    } finally {
      if(cmdInsertPackage != null)
        cmdInsertPackage.Dispose();
    }
  }
}
```

5.5 Lấy dữ liệu bằng SqlCeDataReader

5.5.1 Lấy dữ liệu bằng SqlCeDataReader

Dữ liệu có thể được lấy CSDL SQL CE bằng cách sử dụng lớp SqlCeDataReader. Lớp SqlCeDataReader cung cấp truy nhập nhanh, chỉ một hướng về phía trước tới các bản ghi dữ liệu.

Các bước để nhận dữ liệu bằng SqlCeDataReader như sau:

Bươc 1: Tạo một thể hiện SqlCeConnection. SqlCeDataReader sẽ sử dụng kết nối để nhận dòng dữ liệu yêu cầu.

Bước 2: Đối tượng SqlCeCommand sẽ được tạo cùng vi câu lệnh SELECT thích hợp.

Bước 3: Thiết lập kiểu câu lệnh, và gọi phương thức SqlCeCommand.ExecuteReader.

Phương thức ExecuteReader thực thi command text đối với CSDL bằng SqlCeConnection. SqlCeDataReader sẽ cung cấp truy cập dữ liệu để trả về dữ liệu được trả về. SqlCeConnection sẽ bận will SqlCeDataReader đến khi quá trình đọc dữ liệu đóng lại.

Phương thức đưa đến một tham số của kiểu CommandBehavior. Kiểu CommandBehavior là một tập hợp mà SqlCeCommand sử dụng. Bảng 5.8 là danh sách giá trị của CommandBehavior bà mô tả.

Bảng5.8. Giá trị CommandBehavior

Tên	Mô tả
CloseConnection	Kết nối được đóng lại sau khi đọc dữ liệu được đóng.
Default	Truy vấn có thể trả về nhiều tập kết quả.
KeyInfo	Truy vấn trả về thông tin của cột và khóa chính. Truy vấn được thực thi mà không có bất kỳ dòng nào lựa chọn bị khóa
SchemaOnly	Truy vấn trả về thông tin của cột.
SequentialAccess	Truy vấn cung cấp một cách cho DataReader thao tác các hàng chứa đựng các cột có giá trị nhị phân lớn.
SingleResult	Truy vấn trả về một tập kết quả đơn.
SingleRow	Truy vấn trả về một dòng. Nó chỉ ra vị trí của SingleRow khi thực thi truy vấn mà kết quả là tập hợp nhiều kết quả. Trong trường hợp này, kết quả trả về là tập nhiều kết quả, mỗi kết quả trả về là một dòng.

Một SqlCeDataReader được trả về dựa vào gọi ExecuteReader. Sự tiến bộ của phương thức là đọc các bản ghi tiếp theo. SqlCeDataReader có vị trí khởi tạo là trước bảng ghi đầu tiền. Vì vậy phải gọi Read trước khi yêu cầu lấy dữ liệu. Phương thức Read sẽ trả về true đến tận khi SqlCeDataReader đến cuối của tập kết quả trả về. Sau đó tả về kết quả false.

Chúng xác định được vị trí dòng dữ liệu, chúng ta có thể sử dụng các phương thức GetXXX của SqlCeDataReader để truy nhập các cột trong mỗi dòng dữ liệu. Phương thức GetInt32 nhận một giá trị Int32 từ một cột trong dòng hiện tại của SqlCeDataReader. Phương thức đưa đến một tham số kiểu int. Tham số này thể hiện số thứ tự của cột. Nếu thứ tự của cột không biết đến khi tiết kế, chúng ta có thể sử dụng phương thức GetOrdinal để tìm số thứ tự của cột bằng tên cột. Trong Listing 5.6 mô tả cách nhận tất cả thông tin từ bảng Package.

Listing 5.6 Nhận tất cả thông tin trong bảng Package

```
public static void GetAllPackageInfo() {
  string pckgStr =
    "Package Data\nID: {0}\nCode: {1}\nDestination: {2}";
  string connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connstr)) {
    conn.Open();

    string dmlPackageInfo = "SELECT * FROM Package";
    SqlCeCommand cmdGetPackageInfo = null;
    SqlCeDataReader drPackageInfo = null;

    try {
      cmdGetPackageInfo = new SqlCeCommand(dmlPackageInfo, conn);
      cmdGetPackageInfo.CommandType = CommandType.Text;
      drPackageInfo =
        cmdGetPackageInfo.ExecuteReader(CommandBehavior.Default);

      while(drPackageInfo.Read()) {
        System.Windows.Forms.MessageBox.Show(
          string.Format(pckgStr,
          drPackageInfo.GetInt32(0),
```

```
          drPackageInfo.GetString(1),
          drPackageInfo.GetString(2)));
      }
    } catch(SqlCeException scee) {
      for(int curExNdx = 0; curExNdx < scee.Errors.Count; ++curExNdx) {
        System.Windows.Forms.MessageBox.Show(
          "Error:"+ scee.Errors[curExNdx].ToString()+"\n");
      }
    } finally {
      if( cmdGetPackageInfo != null )
        cmdGetPackageInfo.Dispose();

      if( drPackageInfo != null )
        drPackageInfo.Close();
    }
  }
}
```

5.5.2 Sử dụng tham số SQL Commands

Câu lệnh `SELECT` sử dụng trong Listing 5.5 rất đơn giản. Trong các câu lệnh `SELECT` sẽ hầy hết sử dụng mệnh đề `WHERE`, cái đó sẽ giúp chúng ta lấy những dòng cần thiết. Chúng ta có thể sử dụng mệnh đề `WHERE` để lựa chọn thông tin trong bảng `Package`. Một ví dụ về truy `SELECT`:

```
SELECT * FROM Package WHERE ID = "0987654321"
```

Truy vấn này `SELECT` lấy về những dòng có cột `ID` có giá trị 0987654321.

Chúng ta hãy tạo một đối tượng `SqlCeCommand`. Đối tượng `SqlCeCommand` cung cấp thuộc tính `Parameters` chứa đựng tập hợp tất cả các tham số. Để thêm tham số vào tập hợp này chúng ta sử dụng phương thức `SqlCeCommand.Prepare`.

Listing 5.7 Thực thi một tham số SQL command

```
public static void GetPackageInfo(int pckgID) {
  string pckgStr =
    "Package Data\nID: {0}\nCode: {1}\nDestination: {2}";
  string connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connstr)) {
    conn.Open();

    string dmlPackageInfo = "SELECT * FROM Package WHERE ID = ?";
    SqlCeCommand cmdGetPackageInfo = null;
    SqlCeDataReader drPackageInfo = null;

    try {
      cmdGetPackageInfo = new SqlCeCommand(dmlPackageInfo, conn);
      cmdGetPackageInfo.CommandType = CommandType.Text;
      cmdGetPackageInfo.Parameters.Add("ID", pckgID);
      cmdGetPackageInfo.Prepare();

      drPackageInfo =
        cmdGetPackageInfo.ExecuteReader(CommandBehavior.SingleRow);

      while(drPackageInfo.Read()) {
        System.Windows.Forms.MessageBox.Show(
          string.Format(pckgStr,
          drPackageInfo.GetInt32(0),
```

```
          drPackageInfo.GetString(1),
          drPackageInfo.GetString(2)));
        }
      } catch(SqlCeException scee) {
        for(int curExNdx = 0; curExNdx < scee.Errors.Count; ++curExNdx) {
          System.Windows.Forms.MessageBox.Show(
            "Error:"+ scee.Errors[curExNdx].ToString()+"\n");
        }
      } finally {
        if( cmdGetPackageInfo != null )
          cmdGetPackageInfo.Dispose();

        if( drPackageInfo != null )
          drPackageInfo.Close();
      }
    }
}
```

Truy vấn có tham số có thể được sử dụng trong hầu hết các câu SQL, DDL và DML. Nó có thể được sử dụng nhiều hơn một tham số trong truy vấn. Ví dụ, truy vấn sau có thể được sử dụng để `SELECT`.

```
SELECT * FROM Package WHERE Code = ? OR DestinationID = ?
```

Khi sử dụng câu lệnh `SELECT` cùng với nhiều tham số, chúng ta phải thêm đối tượng `SqlCeParameters` vào tập hợp `Parameters` theo thứ tự dấu ? xuất hiện từ trái sang phải.

Listing 5.8 Thực thi SQL command cùng với nhiều tham số

```
public static void GetPackageInfo(int[] pckgID) {
  string pckgStr =
    "Package Data\nID: {0}\nCode: {1}\nDestination: {2}";
  string connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connstr)) {
    conn.Open();

    string dmlPackageInfo = "SELECT * FROM Package WHERE ID = ?";
    SqlCeCommand cmdGetPackageInfo = null;
    SqlCeDataReader drPackageInfo = null;

    try {
      cmdGetPackageInfo = new SqlCeCommand(dmlPackageInfo, conn);
      cmdGetPackageInfo.CommandType = CommandType.Text;
      cmdGetPackageInfo.Parameters.Add("ID", SqlDbType.Int);
      cmdGetPackageInfo.Prepare();

      for(int pckgNdx = 0; pckgNdx < pckgID.Length; ++pckgNdx) {
        cmdGetPackageInfo.Parameters[0].Value = pckgID[pckgNdx];
        try {
          drPackageInfo =
            cmdGetPackageInfo.ExecuteReader(CommandBehavior.SingleRow);

          while(drPackageInfo.Read()) {
            System.Windows.Forms.MessageBox.Show(
              string.Format(pckgStr,
              drPackageInfo.GetInt32(0),
              drPackageInfo.GetString(1),
              drPackageInfo.GetString(2)));
          }
        } catch(SqlCeException scee) {
```

```
          for(int curExNdx=0;curExNdx<scee.Errors.Count;++curExNdx) {
            System.Windows.Forms.MessageBox.Show(
              "Error:"+ scee.Errors[curExNdx].ToString()+"\n");
          }
        } finally {
          if( drPackageInfo != null )
            drPackageInfo.Close();
        }
      }
    } finally {
      if( cmdGetPackageInfo != null )
        cmdGetPackageInfo.Dispose();
    }
  }
}
```

5.6 Lọc một `DataSet` bằng `SqlCeDataAdapter`

Compact Framework cung cấp khả năng lập dữ liệu trực tiếp từ SQL Server CE vào một `DataSet`. Điều này được hoàn thành bằng cách sử dụng `SqlCeDataAdapter` đưa vào `DataSet`. `SqlCeDataAdapter` có thể đưa vào `DataSet` và cập nhật vào CSDL. `DataSet` có thể quản lý tất cả các giao tiếp giữa ứng dụng và CSDL SQL Server CE .

Quản lý `SqlCeDataAdapter` trong CSDL bằng cách chạy các câu lệnh khác nhau. Có bốn câu lệnh được đưa ra như là thuộc tính trên `SqlCeDataAdapter`, đó là `SelectCommand`, `InsertCommand`, `UpdateCommand`, và `DeleteCommand`.

Thuộc tính `SelectCommand` là đối tượng `SqlCeCommand` xác định là câu lệnh SQL mà `SqlCeDataAdapter` sẽ sử dụng để nhậ dữ liệu từ CSDL SQL Server CE database. `SqlCeDataAdapter` sẽ sử dụng dữ liệu để đưa vào `DataSet`.

Bao gồm các bước sau:

Bước 1: Xây dựng một `DataSet`

Bước 2: Nhận dữ liệu

Bước 3: Đưa vào `DataSet`

Trước tiên, `SqlCeDataAdapter` khởi tạo giản đồ `DataSet` tương ứng với giản đồ trong nguồn dữ liệu, Điều này có nghĩa là `DataTable`s được xây dựng tương ứng với bảng CSDL nguồn như là xây dựng `DataColumn`s tương ứng với cột bảng CSDL nguồn. Quan hệ giữa `DataSet` và CSDL nguồn được biết như là ánh xạ bởi vì chúng ánh xạ đối tượng `DataSet` vào đối tượng CSDL. Tiếp theo dữ liệu được nhận về từ CSDL nguồn bằng cách sử dụng thuộc tính `SelectCommand`. Cuối cùng `DataRows` được tạo để nhận dữ liệu, và các dòng được trèn vào `DataTable`s.

Sau đây là đoạn mã đưa dữ liệu vào một `DataSet` bằng cách sử dụng `SqlCeDataAdapter` rất đơn giản. Listing 5.9 mô tả cách đưa dữ liệu của bảng `Package` vào `DataSet` bằng cách sử dụng `SqlCeDataAdapter`.

Listing 5.9 Đưa dữ liệu vào `DataSet` cùng với nội dung của bảng `Package`

```
public static DataSet GetPackageDataSet() {
```

```
  string connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connstr)) {
    conn.Open();

    string dmlPackageInfo = "SELECT * FROM Package";
    SqlCeDataAdapter daPackages = new SqlCeDataAdapter();
    daPackages.MissingMappingAction = MissingMappingAction.Passthrough;
    daPackages.MissingSchemaAction = MissingSchemaAction.Add;
    daPackages.SelectCommand = new SqlCeCommand(dmlPackageInfo, conn);

    DataSet dsPackages = new DataSet();
    daPackages.Fill(dsPackages);

    return dsPackages;
}
```

Trước tiên tạo và mở một kết nối tới CSDL SQL Server CE. Sau đó tạo một đối tượng `SqlCeDataAdapter` và thiết lập `MissingMappingAction` và thuộc tính `MissingSchemaAction`. Thiết lập thuộc tính mặc định. Tiếp theo, thiết lập `SelectCommand` thành một đối tượng `SelectCommand` lựa chọn tất cả dữ liệu từ bảng `Package`. Cuối cùng, tạo đối tượng `DataSet` và gọi phương thức `SqlCeDataAdapter.Fill` để đưa dữ liệu vào `DataSet` với dữ liệu trong bảng `Package`.

5.7 Cập nhật CSDL Microsoft SQL Server CE sử dụng `SqlCeDataAdapter`

`DataSet` đã đưa dữ liệu vào bằng cách sử dụng `SqlCeDataAdapter`, chúng ta có thể tạo sự thay đổi dữ liệu và cập nhật dữ liệu nguồn, chúng ta phải chỉ ra ba thuộc tính thêm vào đối tượng `SqlCommand` cho `SqlCeDataAdapter` là: `UpdateCommand`, `InsertCommand`, và `DeleteCommand`.

Listing 5.11 Sử dụng `SqlCeDataAdapter` để cập nhật dữ liệu

```
public static
SqlCeDataAdapter GetPackageDataAdapter(SqlCeConnection conn){
  string dmlPackageInfo = "SELECT * FROM Package";
  string dmlUpdatePackage="UPDATE Package " +
                         "SET CODE = ?, " +
                         "    DestinationID = ? " +
                         "WHERE ID = ?";
  string dmlInsertPackage="INSERT INTO " +
                         "Package(Code, DestinationID) " +
                         "VALUES (?, ?)";
  string dmlDeletePackage="DELETE FROM " +
                         "Package " +
                         "WHERE ID = ?";

  SqlCeDataAdapter daPackages = new SqlCeDataAdapter();
  daPackages.SelectCommand = new SqlCeCommand(dmlPackageInfo, conn);

  daPackages.UpdateCommand = new SqlCeCommand(dmlUpdatePackage, conn);
  daPackages.UpdateCommand.Parameters.Add("Code", SqlDbType.NVarChar);
  daPackages.UpdateCommand.Parameters.Add("DestinationID",
    SqlDbType.NVarChar);
  daPackages.UpdateCommand.Parameters.Add("ID", SqlDbType.Int);
```

```
  daPackages.InsertCommand = new SqlCeCommand(dmlInsertPackage, conn);
  daPackages.InsertCommand.Parameters.Add("Code", SqlDbType.NVarChar);
  daPackages.InsertCommand.Parameters.Add("DestinationID",
    SqlDbType.NVarChar);

  daPackages.DeleteCommand = new SqlCeCommand(dmlDeletePackage, conn);
  daPackages.DeleteCommand.Parameters.Add("ID", SqlDbType.Int);

  return daPackages;
}
```

SqlCeDataAdapter cập nhật dữ liệu nguồn khi chúng ta gọi phương thức Update. Phương thức Update thao tác qua 5 bước khi cập nhật dữ liệu:

Bước 1: Các giá trị cập nhật được nạp vào từ đối tượng DataRow trong tham số câu lệnh có liên quan.

Bước 2: Sự kiện RowUpdating được đưa ra.

Bước 3: Câu lệnh liên quan được thực thi đối với dữ liệu nguồn.

Bước 4: Sự kiện RowUpdated được đưa ra.

Bước 5: Thuộc tính RowSet của DataRow được thiết lập lại RowState.Unchanged bằng cách gọi phương thức AcceptChanges.

Listing 5.12 Cập nhật bảng `Package` sử dụng `SqlDataAdapter`

```
public static void UpdatePackageTable(DataSet dsPackages) {
  string connstr = @"Data Source=\My Documents\PTSystem.sdf";

  using(SqlCeConnection conn = new SqlCeConnection(connstr)) {
    conn.Open();

    SqlCeDataAdapter daPackages = GetPackageDataAdapter(conn);
    daPackages.Update(dsPackages);
  }
}
```

Thao tác với sự kiện cập nhật SqlCeDataAdapter

Khi chúng ta gọi phương thức cập nhật trên SqlCeDataAdapter, có hai sự kiện được đưa ra. Sự kiện RowUpdating được đưa ra trước câu lệnh Update được thực thi với dữ liệu nguồn. Sự kiện RowUpdated được phát sinh sau khi câu lệnh Update được thực thi với dữ liệu nguồn.

Khi chúng ta nhận một sự kiện RowUpdating, chúng ta sẽ xác định thuộc tính của đối tượng SqlCeRowUpdatingEventArgs và quyết định tiếp tục cập nhật hay không.

5.8 Đối tượng SqlCommand với SqlCeCommandBuilder

Trước tiên, chúng ta cần khởi tạo SqlCeDataAdapter và thuộc tính SelectCommand. Sau đó chúng ta tạo SqlCeCommandBuilder thông qua SqlCeDataAdapter như là tham số để cấu trúc SqlCeCommandBuilder. SqlCeCommandBuilder sẽ tạo một câu lệnh cho thuộc tính UpdateCommand, InsertCommand, and DeleteCommand của SqlCeDataAdapter. Listing 5.14 mô

tả cách sử dụng SqlCeCommandBuilder để xây dựng một SqlCeDataAdapter cho bảng Package.

Listing 5.14 Sử dụng SqlCeCommandBuilder

```
public static
SqlCeDataAdapter GetPackageDataAdapter(SqlCeConnection conn){
  string dmlPackageInfo = "SELECT * FROM Package";
  SqlCeDataAdapter daPackages = new SqlCeDataAdapter();
  daPackages.SelectCommand = new SqlCeCommand(dmlPackageInfo, conn);

  SqlCeCommandBuilder cmdBldr = new SqlCeCommandBuilder(daPackages);
  MessageBox.Show(cmdBldr.GetUpdateCommand().CommandText);
  MessageBox.Show(cmdBldr.GetInsertCommand().CommandText);
  MessageBox.Show(cmdBldr.GetDeleteCommand().CommandText);

  return daPackages;
}
```

Chương 6 Phát triển cho SmartPhone

6.1 Giới thiệu SmartPhone

SmartPhone về bản chất là cell phone chạy hệ điều hành Pocket PC operating system. Để thuận tiện khi làm việc với các thiết bị nhỏ, SmartPhone trênh lệch so với chuẩn Pocket PC có hai cách quan trọng sau:

- Kích cỡ màn hình cho SmartPhone là nhỏ hơn so với các thiết bị chuẩn Pocket PC. Độ phân giải màn hình SmartPhone là 176 x 220, so sánh với chuẩn Pocket PC là 240 x 320.

- Màn hình SmartPhone không dễ hỏng. Thay đổi cơ bản kiểu dáng cái mà người sử dụng đưa thông tin vào ứng dụng. Người sử dụng tương tác cùng với ứng dụng bằng các nút vật lý trên điện thoại.

Có hai cách khác nhau để phát triển cho SmartPhones. Sự khác biệt chủ yếu là màn hình nhỏ, yêu cầu người phát triển phải quan tâm đến màn hình thực cẩn thận. Nhưng thiếu màn hình sờ và bàn phím có nghĩa là

Trong phần này học cách phát triển cho SmartPhone bằng .NET Compact Framework trong khi làm việc xung quanh SmartPhone có sự hạn chế cố hữu của nó.

6.2 Phát triển SmartPhone bằng .NET Compact Framework

Để phát triển cho SmartPhone, chúng ta phải cài đặt gói hỗ trợ cho Visual Studio. Cái này có sẵn ở Microsoft. Mặc định, Smart Device Extensions cho Visual Studio cho phép chúng ta tạo các dự án cho nền tảng Pocket PC hoặc Windows CE. Để thêm gói hỗ trợ cho SmartPhone, nền tảng SmartPhone được thêm vào như một kiểu dự án.

Bởi vì SmartPhone add-on đơn giản là mở rộng cho Smart Device Extensions, những nhà phát triển có kinh nghiệm khi phát triển cho nền tảng SmartPhone như khi phát triển cho Pocket PC hoặc Windows CE. Khác nhau chính là có những emulators để deploy và điều khiển không hỗ trợ SmartPhone. Một số điều khiển sau không hỗ trợ:

- `Button`
- `RadioButton`
- `ListBox`
- `TabControl`
- `DomainUpDown`
- `NumericUpDown`
- `TrackBar`
- `ContextMenu`
- `ToolBar`

- `StatusBar`
- `OpenFileDialog`
- `SaveFileDialog`
- `InputPanel`

Khó khăn chính trong làm việc với SmartPhone là thiết kế một giao diện người sử dụng sử dụng các điều khiển có sẵn trong hộp công cụ ToolBox. Khả năng còn lại của .NET Compact Framework vẫn sẵn sàng cho ứng dụng của chúng ta.

Đưa ra hệ điều hành SmartPhone hỗn hợp

Nó rất quan trọng để nhận thức về sự khác nhau trong hệ thống file SmartPhone so với nền tảng Pocket PC và Windows CE đầy đủ. Trên SmartPhone chỉ có thư mục `\Storage`. Tất cả ứng dụng quản lý được đưa vào thưc mục `\Storage`, và chúng ta có thể tạo file vào thư mục `\Storage`.

6.3 Viết một ứng dụng cho SmartPhone - XMLDataSetViewer

Chúng ta sẽ tiếp cận xậy dựng khung nhìn đơn giản XML `DataSet`. Khung nhìn XML `DataSet` đưa đến bảng đầu tiên trong `DataSet`. Hướng dẫn cung cấp cho người phát triển cùng với dự án SmartPhone đã tồn tại để thử nhiệm. Nó mô tả cách .NET Compact Framework phát triển có kinh nghiệm gần như không thay đổi khi làm việc với SmartPhone.

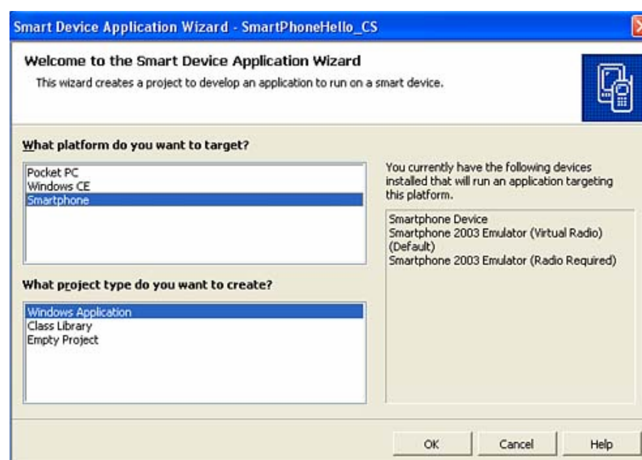
Trước khi bắt đầu, chúng ta coi như SmartPhone add-on đã cài đặt. Các bước như sau:

Xây dựng DataSetViewer:

Bước 1: Chạy Visual Studio .NET và tạo mới một dự án. Chúng ta có thể chọn một ứng dụng Smart Device bằn C#.

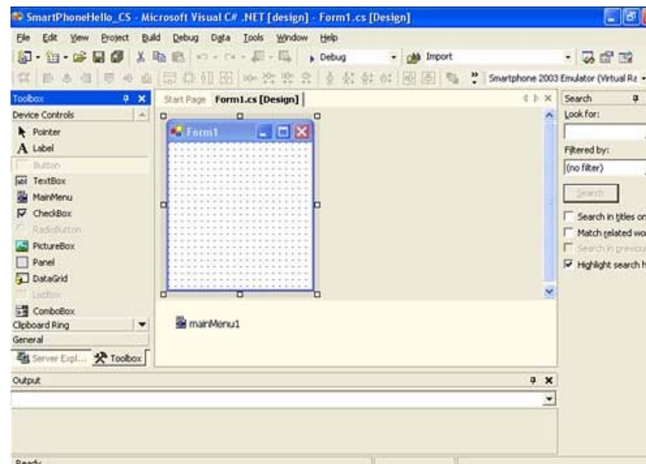
Bước 2: Sau đó các bước như chúng ta thao tác tạo một ứng dụng Smart Device, chấp nhận khi hỏi chấp nhận nền tảng, chọn SmartPhone của Pocket PC hoặc Windows CE. Bước này được đưa đến trong hình hình 6.1.

Hình 6.1. Khi tạo một dự án mới, chọn nền tảng SmartPhone



Bước 3: Khi chúng ta kết thúc thiết lập ứng dụng mới, chúng ta sẽ sem phần sửa form và Toolbox, như trong hình 6.2. Hình cho thấy hầu hết những cái chính của dự án Pocket PC, ngoại trừ form nhỏ hơn để mang lại cho màn hình nhỏ trên SmartPhones. Mặc dù một số điều khiển trong Toolbox bị mờ đi.

Hình 17.2. Sửa form và hộp công cụ cho dự án SmartPhone chứa đựng một số điều khiển bị mờ đi.



Bước 4: Kéo một DataGrid và một TextBox vào form. Tên của DataGrid là `dgDataSet` và tên của TextBox là `txtXmlToLoad`. Sử dụng giá trị mặc đinh cho TextBox là: `\Storage\Program Files\XMLDataSetViewer_CS\SampleDataSet.xml`.

Bước 5: Thêm một tham chiếu đến DataGrid. Để làm điều này, bấm chuột phải vào tên solution (`XmlDataSetView_CS`) trong Solution Explorer. Sau đó chọn Add Reference. Chúng ta sẽ nhìn thấy hộp thoại trong đó có thể chọn rất nhiều các DLLs. Chọn nút Browse và di chuyển tới thư mục trong thư mục cài Visual Studio (`C:\Program Files\Microsoft Visual Studio .NET 2003`). Trong thư mục lựa chọn file `CompactFrameworkSDK\v1.0.5000\Windows CE\System.Windows.Forms.DataGrid.dll`.

Bước 6: Thêm đối tượng menu bằng cách chọn biểu tượng `MainMenu1` xuất hiện dưới phần sửa form (form editor) trong IDE. Chúng ta có thể thêm các mục trong menu bằng cách bấm khe thêm menu mới sau đó gõ text cho menu. Ví dụ: `Exit` và `Load XML`.

Bước 7: Thêm mã lệnh cho menu `Exit` bằng cách bấm đúp vào nó. IDE mang đếm phương thức nhận được gọi khi menu `Exit` được chọn. Thêm mã lệnh:

```
Application.Exit();
```

Bước 8: Thêm biến cho `DataSet`, có tên là `m_DataSet`, ở trên cùng của lớp `Form1`. Ví dụ, biến thành viên của lớp cho dự án như sau:

```
public class Form1 : System.Windows.Forms.Form
{
   private System.Windows.Forms.DataGrid dgDataSet;
   private System.Windows.Forms.MenuItem menuItem1;
   private System.Windows.Forms.MenuItem menuItem2;
```

```
    private System.Windows.Forms.TextBox txtXmlToLoad;
    private System.Windows.Forms.MainMenu mainMenu1;

    private DataSet m_DataSet;
// Rest of class Form1 not shown here...
```

Bước 9: Thêm mã lệnh cho menu `Load XML` bằng cách bấm dúp chuột vào mục đó và đưa vào đoạn mã lệnh như sau:

```
if (this.m_DataSet == null)
{
   this.m_DataSet = new DataSet();
}
this.m_DataSet.Clear();
try
{
   m_DataSet.ReadXml(this.txtXmlToLoad.Text);

   // Set up a DataView
   DataView l_DataView = new DataView(m_DataSet.Tables[0]);
   this.dgDataSet.DataSource = l_DataView;
}
catch (Exception ex)
{
   MessageBox.Show(ex.ToString());
}
```

Bước 10: Thêm file mặc đinh XML, `SampleDataSet.xml`, vào ứng dụng. Để làm điều này, trong Solution Explorer bấm `Ctrl+Alt+L` và đưa chuột qua tên các solution (ví dụ, `XMLDataSetViewer_CS`). Bấm chuột phải và chọn `Add`, `Add Existing Item`, và sau đó chọn file `SampleDataSet.xml`. Chúng ta có thể tìm file này trong dự án XMLDataSetViewer.

Bước 11: Xây dựng và triển khai ứng dụng! Nếu chúng ta không có bất kỳ thiết bị SmartPhone, có thể triển khai bằng các emulator trong Virtual Radio.

Sử dụng XML DataSetViewer

Sử dụng ứng dụng khác với sử dụng ứng dụng trên Pocket PC bởi vì không bàn phím sờ. Không có bàn phím, và một số nút.

Để chọn file XML để nạp file vào ứng dụng, chúng ta phải trên đầy đủ đường dẫn của file XML vào textbox. Để làm điều này, trước tiên tạo textbox vào ứng dụng. Một textbox được kích hoạt, chúng ta có thể di chuyển con trỏ cùng với con trỏ và trên văn bản vào bằng phím số.

Để nạp file XML, bấm menu phải, mục `Load XML`

Chương 7 Sử dụng XML Web Services

7.1 Tạo XML Web Service

Trước khi tạo .NET Compact Framework XML Web service client, XML Web Service client phải sử dụng được. Trong phần này chúng ta tạo một Web service, và tạo một a .NET Compact Framework client để sử dụng Web service. XML Web service trả về thông tin của một cá nhân.

Thông tin của người đó được lưu trong CSDL Microsoft SQL Server. Khi một yêu cầu được tạo ra, XML Web service sẽ truy vấn một trích dẫn ngâu nghiên và trả về thông tin trích dẫn. Chúng ta cần phải thiết lập CSDL này trước khi chúng ta có thể chạy ví dụ này.

Để tạo XML Web service trong Visual Studio.NET, sử dụng ASP.NET Web Service template. Tên dự án là `QuotableQuotesWebService`. XML Web service có tên là `Service1` sẽ được tạo file `Service1.asmx`. Thay đổi tên XML Web service thành `QuoteService` và file nguồn `.aspx` là `QuoteService.aspx`.

XML Web service sẽ đưa ra một phương thức web, `GetQuote`. Phương thức này trả về thông tin trích dẫn. Thông tin trích dẫn được lấy từ CSDL Microsoft SQL Server. Có một thủ tục trong CSDL `QuotableQuotes` có tên là `GetQuote`, thủ tục này chúng ta sẽ sử dụng để truy vấn thông tin trích dẫn. Microsoft Visual Studio.NET sẽ trợ giúp trong quá trình viết mã lệnh để tác động đến thủ tục này. Trước tiên mở Server Explorer và tạo stored procedure `GetQuote` trong CSDL `QuotableQuotes`. Kéo stored procedure `GetQuote` vào trong phần thiết kế XML Web service. Chúng ta tạo hai đối tượng: `sqlConnection1` và `sqlCommand1`. Đối tượng `sqlConnection1` có kiểu `SqlConnection` và thể hiện kết nối tới CSDL `QuotableQuotes`. Đối tượng `sqlCommand1` có kiểu là `SqlCommand` và thể hiện SQL command sẽ nhận thông tin trích dẫn từ stored procedure. Đổi tên `sqlConnection1` và `sqlCommand1` lần lượt thành `quoteConnection` và `cmdGetQuote`.

Trước khi thực thi `GetQuote`, cần phải có phương thức giúp đỡ để tạo thông tin trích dẫn ngâu nghiên. SqlCommand `cmdGetQuote` đưa đến một tham số. Tham số này là ID của bản ghi thông tin trích dẫn trong CSDL. Trong CSDL mỗi thông tin trích dẫn có một trường khóa có kiểu `integer`. Trường khóa này tự động tăng, mỗi lần tăng nên một, và giá trị đầu tiền là 0. `QuotableQuote` XML Web service sẽ trả về thông tin trích dẫn ngẫu nghiên. Để làm điều này, trong mã nguồn phải tạo một số ngẫu nghiên từ 0 và giá trị lớn nhất của trường khóa trong CSDL. Số lớn nhất đó phải nhận từ CSDL. Có một stored procedure có tên là `GetLargestQuoteIdentifier` trong CSDL làm điều này. Đặt `GetLargestQuoteID` vào stored procedure, và kéo vào phần thiết kế. Nó sẽ tạo một đối tượng `SqlCommand`. Đổi tên thành `cmdGetLargestID`. Đoạn mã sau mô tả cách nhận giá trị trường ID lớn nhất từ CSDL. Đoạn mã này sẽ ở trong lớp `QuoteService`.

Listing 7.1

```
public Int64 LargestID
{
  get
  {
    object largestID = cmdGetLargestID.ExecuteScalar();
    if(largestID== null || !(largestID is Int64))
      return -1;

    return (Int64)largestID;
  }
}
```

Trước khi viết mã lệnh để nhận giá trị lớn nhất của trường khóa từ bảng `Quotes`. Trước tiên, đối tượng `cmdGetLargestID SqlCommand` được sử dụng để nhận giá trị lớn nhất trường khóa từ CSDL. Khi giá trị nhận về được kiểm tra đúng. Giá trị -1 được trả về nếu giá trị không hợp lệ.

Sau khi nhận giá trị của trường khóa lớn nhất, một giá trị IP ngâu nghiên được tạo. Làm điều này cùng với lớp `System.Random`. Lớp `System.Random` thể hiện tạo một số ngẫy nghiên. Phương thức `Next` sẽ được sử dụng để nhận một số nguyên ngâu nghiên (`Int32`). Phương thức `Next` có thể chấp nhận một số nguyên (`Int32`), số này thể hiện giới hạn trên của số ngẫu nghiên để phát sinh. Trong ví dụ này giá trị lớn nhất của ID sẽ được tạo được thông qua như là một tham số.

Phương thức sẽ trả về cấu trúc dữ liệu khách hàng, cấu trúc này chứa đựng thông tin trích dẫn. Listing 7.2 chứa đựng lớp `Quote`, lớp này lưu trữ thông tin trích dẫn. Lớp sẽ đực đặt trong file `QuoteService.aspx` bên trong của không gian tên.

Listing 7.2

```
public class Quote
{
  public string String;
  public string Author;
  public string Date;
}
```

Phương thức `GetQuote` phải được thực thi. Phương thức `GetQuote` Web cần phải hoàn thành những công việc sau:

Bước 1: Phát sinh một giá trị `Quote` ID ngẫy nghiên.

Bước 2: Lấy dữ liệu trích dẫn từ CSDL.

Bước 3: Điền vào cấu trúc dữ liệu `Quote`.

Bước 4: Trả về cấu trúc dữ liệu `Quote`.

Trong đoạn mã Listing 7.3 đưa đến phương thức Hello World trong file `QuoteService.aspx`.

Listing 7.3

```
[WebMethod]
```

```
public Quote GetQuote()
{
  quoteConnection.Open();

  try
  {
    Int64 largestID = LargestID;
    if(-1 == largestID)
      return null;

    Random rand = new Random(DateTime.Now.Millisecond);
    Int64 randomQuoteId = rand.Next((int)largestID);
    cmdGetQuote.Parameters["@id"] =
      new SqlParameter("@id", randomQuoteId);

    SqlDataReader reader = cmdGetQuote.ExecuteReader();
    if(!reader.Read())
      return null;

    Quote q = new Quote();
    q.String = reader.GetString(0);  // Get Quote String
    q.Author = reader.GetString(1);  // Get author's name
    q.Date = reader.GetString(2);    // Get the spoken date

    return q;
  }
  finally
  {
    quoteConnection.Close();
  }
}
```

Trước tiên kết nối với CSDL `QuotableQuotes` đã được mở bằng phương thức `Open` trên đối tượng `quoteConnection`. Tiếp theo một giá trị ngẫy nghiên giữa 0 và giá trị lớn nhất được phát sinh bằng phương thức `Next` trên lớp `System.Random`. ID được kiểm tra tính hợp lệ. Nếu ID hợp lệ, giá trị đó được thiết lập như là tham số có tên là `@id` của đối tượng `cmdGetQuote SqlCommand`. Tiếp theo phương thức `ExecuteReader` của đối tượng `SqlComman` được gọi. Phương thức này thực thi câu lệnh đối với CSDL Microsoft SQL Server và trả về một đối tượng `SqlDataReader`, đối tượng này cung cấp truy cập vào dữ liệu trích dẫn. Sau đó `SqlDataReader` điền vào cấu trúc dữ liệu `Quote`. Cuối cùng, cấu trúc dữ liệu `Quote` được trả về, và khối `finally` đảm bảo rằng kết nối CSDL được đóng trong trường hợp có ngoại lệ. Trước khi lớp trong không gian tên `SqlClient` có thể được sử dụng, không gian tên `System.Data.SqlClient` phải được đưa vào trong file `QuoteService.aspx`.

Mặc định , một Web service mới được đưa vào không gian tên `http://tempura.org`. Microsoft khuyến cáo rằng mỗi XML Web service có một không gian tên XML duy nhất. Điều này cho phép ứng dụng client chỉ ra sự khác biệt nó với các dịch vụ khác trên Web. Có thể hoàn thành bằng cách áp dụng thuộc tính `WebServiceAttribute` đối với lớp Web service. Thêm các dòng lệnh sau vào lớp `QuoteService`:

```
[WebService(Namespace="http://netcfkickstart/QuoteService",
            Description="Provides access to famous quotes")]
```

Thay đổi thuộc tính không gian tên của `QuoteService` như là thêm một mô tả ngắn gọn về Web service.

Cùng với thuộc tính `WebServiceAttribute` ứng dụng vào Web service, QuotableQoutes Web service có thể được dịch và thử. Nhấn phím F5 để dịch và gỡ lỗi XML Web service. `QuoteService` đưa đến trang Web. Trang Web này chứa đựng chuỗi mô tả từ thuộc tính `WebServiceAttribute`. Trang chứa một nhãn liên kết `Service Description`. Liên kết này sẽ hiển thị định dạng file WSDL cho dịch vụ. Có liên kết cùgn với text `GetQuote`. Chọn liên kết đó sẽ đưa đến trang Web cho phép thử phương thức `GetQuote` Web.

Trang kiểm thử này cung cấp một vài thông tin. Chọn vào nút `Invoke` trên trang Web sẽ gọi phương thức Web và hiển thị thông tin trả về trong Internet Explorer. Sai đây XML là một ví dụ về kết quả trả về từ trang `GetQuote`:

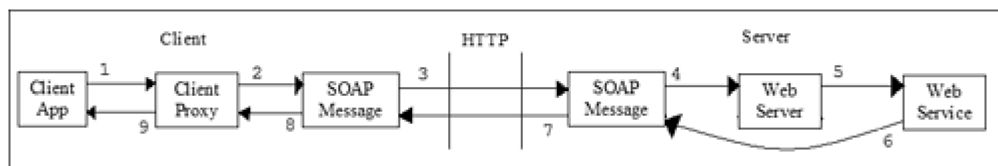
```
<?xml version="1.0" encoding="utf-8" ?>
<Quote xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns="http://netcfkickstart/QuoteService">
  <String>
   "Once you eliminate the impossible, whatever remains, no matter how
   improbable, must be the truth."
   </String>
   <Author>Sherlock Holmes</Author>
          <Date>1859-1930</Date>
   </Quote>
```

Bên cạnh sự cung cấp khả năng kiểm thử Web service, trang này còn cung cấp ba ví dụ về cách yêu cầu Web service và ví dụ trả về. Ví dụ này bao gồm định dạng cho `HTTP-POST`, `HTTP-GET`, và `SOAP`.

7.2 Tìm hiểu .NET Framework Web Service Client

Sử dụng giao thức HTTP và thông điệp SOAP, Web service client có thể gửi yêu cầu và thể hiện thông tin trả về từ Web server. Hình 7.1 mô tả thời gian tồn tại của một phương thức Web gọi từ client tới Web server.

Hình 7.1. Thời gian tồn tại của phương thức Web gọi một Web service.



Sau đây là danh sách mô tả 9 bước trong thời gian tồn tại của phương thứcWeb method.

Bước 1: Ứng dụng client tạo một thể hiện lớp proxy. Proxy là lớp thông điệp giao tiếp giữa ứng dụng client và Web service. Ứng dụng client gọi một phương thức trên lớp proxy

Bước 2: Proxy, sử dụng kiến trúc Web service bên phía client, tuần tự hóa XML Web service yêu cầu và đóng gói nó trong thông điệp SOAP.

Bước 3: Thông điệp SOAP sau đó gửi tới Web server thông qua HTTP.

Bước 4: Kiến trúc Web service phía server nhận và hủy tuần tự hóa thông điệp SOAP.

Bước 5: Sử dụng thông điệp SOAP, lớp thể hiện Web service được tạo. Tiếp theo, đối số của phương thức Web được tạo từ thông điệp SOAP. Cuối cùng, phương thứ Web được gọi cùng với đối số riêng.

Bước 6: Kiến trúc Web service phía server đóng gói đưa ra tham số và trả về giá trị vào thông điệp SOAP.

Bước 7: Thông điệp SOAP được gửi trở lại client thông qua HTTP.

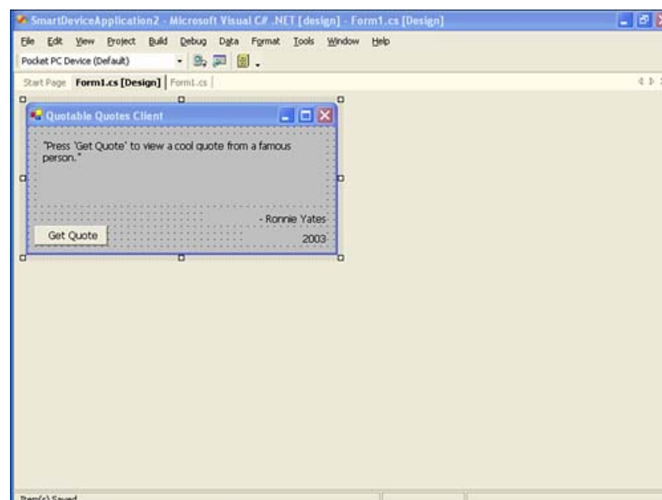
Bước 8: Kiến trúc Web service phía client nhận thông điệp SOAP và hủy tuần tự hóa tham số đưa ra và trả về giá trị. Giá trị được trả về lớp proxy.

Bước 9: Lớp proxy trả về tham số đầu ra và trả về giá trị tới ứng dụng client.

7.3 Tạo một ứng dụng Client XML Web Service.

Bây giờ chúng tạo một client cho `QuotableQuotes` XML Web service. Bắt đầu tạo một ứng dụng Smart Device Application. Thiết kế giao diện độ họa giống hình 7.2. Mã lệnh để gọi XML Web service sẽ được đưa vào trong sự kiện click trên nút có nhãn `Get Quote`. Trước tiên thêm một tham chiếu đên XML Web service trong dự án Smart Device Application.

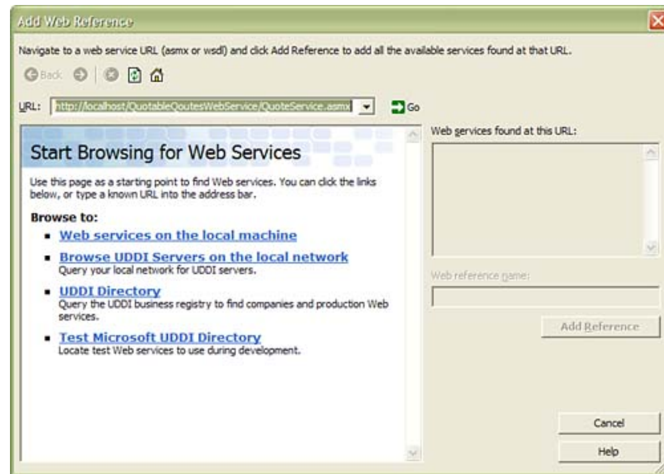
Hình 7.2. Giao diện người sử dụng `QuotableQuotes` client application.



7.3.1 Thêm Web Reference vào Client Application

Bây giờ, Web tham chiếu đến dịch vụ cần thiết thêm vào dự án client. Để làm điều này, vào Solution Explorer, bấm chuột phải vào mục `Reference`, và chọn `Add Web References` .... Hộp thoại Add Web Reference như hình (hinhg 7.3).

Hình 7.3. Hộp thoại Web Reference.



Hộp thoại Add Web Reference giúp xác định Web Services và sử dụng chúng trong ứng dụng. Hộp thoại hiển thị bốn liên kết. Một liên kết cho phép chúng ta xem tất cả XML Web services trên máy. Ba liên kết khác liên kết đến ba UDDI servers/directories. Liên kết "Browse UDDI Servers on the local machine" cho phép chúng ta duyệt UDDI servers trên mạng cục bộ. hai liên kết khác có nhãn là UDDI Directory và UDDI Directory cung cấp truy cập tới dịch vụ đã được đăng ký với Microsoft. Liên kết "UDDI Directory" duyệt UDDI business registry để tìm công ty và sản phẩm Web services. Liên kết "Test Microsoft UDDI Directory" xác định XML Web services kiểm thử, có nghĩa là được sử dụng trong quá trình phát triển.

Trong ví dụ này `QuotableQuotes` XML Web service sẽ không đăng ký với UDDI. Thay vì đăng ký `QuotableQuotes` XML Web service cùng với UDDI, địa chỉ trang `QuoteService.aspx` sẽ được đưa vào trong hộp thoại Add Web Reference. Đưa vào địa chỉ URL sau, `http://localhost/QuotableQuotesWebService/QuoteService.asmx`, vào trong hộp text `Address` và bấm `Enter` hoặc nút `Go`.

Hộp thoại sẽ download file WSDL cho QuoteService Web service. Quá trình download được hoàn thành, bấm vào nút Add Reference.

7.3.2 Xem lớp Proxy

Khi thêm nút Add Reference được bấm, Microsoft Visual Studio.NET phát sinh một lớp proxy, lớp này sẽ quản lý tương tác giữa ứng dụng `QuotableQuotes` Web service. Trong một số trường hợp file class proxy có thể cần hiển thị hoặc thay đổi, nhưng mặc định Solution Explorer không hiển thị file class proxy. Có thể thay đổi bằng cách chọn nút Show All Files trong Solution Explorer. Bây giờ mở rộng nút Web References và nút bên dưới Reference.map. Hiện ra một nút có nhãn Reference.cs. Bấm đúp vào nút đó sẽ hiển thị mã nguồn của lớp proxy.

Có một vài thứ sẽ được gọi từ lớp proxy. Trước tiên, lớp proxy đưa đến khả năng mã hóa client để chỉ ra URL sẽ sử dụng khi giao tiếp với XML Web service. Lớp proxy có thuộc tính `URL` kiểu chuỗi thể hiện địa chỉ URL trang `.aspx` của XML Web service. Từ khi đó Windows CE và Pocket PC emulators có một địa chỉ IP khác tất cả các địa chỉ máy khác đang chạy, chúng ta sẽ không có khả năng xác định XML Web service bằng cách sử dụng địa chỉ URL mặc định trong lớp proxy bằng cách sử dụng localhost như là tên server. Thay vì, sử dụng tên server hoặc

địa chỉ IP, trong phần này chúng ta nên sử dụng địa chỉ IP để ngăn ngừa lỗi cùng với giải pháp tên server.

Thứ hai, trong file lớp proxy, tồn tại một lớp có tên `Quote`. Lớp này tương tự với lớp `Quote` đã được tạo trên phía server để lưu dữ thông tin trích dẫn. Trong Listing 7.4 chứa đựng khai báo lớp `Quote` từ file mã nguồn của lớp proxy.

Listing 7.4

```
[System.Xml.Serialization.XmlTypeAttribute(
Namespace="http://netcfkickstart/QuoteService")]
public class Quote {
  /// <remarks/>
  public string String;

  /// <remarks/>
  public string Author;

  /// <remarks/>
  public string Date;
}
```

Thuộc tính `System.Xml.Serialization.XmlTypeAttribute` chỉ ra kiến trúc XML Web service phía client để sử dụng không gian tên `http://netcfkickstart/QuoteService` khi tuần tự hóa một đối tượng có kiểu này. Lớp `Quote` phải khai báo trong proxy, vì vậy client và server có thể sử dụng cấu trúc dữ liệu giống nhau khi tương tác. Tất cả cấu trúc dữ liệu khách hàng được đưa ra qua đường XML Web service sẽ được khai báo để sử dụng bằng client trong file mã nguồn lớp proxy.

7.3.3 Sử dụng QuotableQuotes Web Service

Bây giờ chúng thêm một dự án client và lớp proxy đã được tạo ra, XML Web service có thể sử dụng bằng ứng dụng clietn. Mở phần thiết kế cho giao diện đồ họa client, và bấm đúp vào nút `Get Quote`. Sau đó hiển thị phần nhập mã lệnh cho sự kiện click của nút, Trước khi thực thi phương thức này, không gian tên của lớp proxy cần phải được thêm vào `Form1.cs`. Lớp proxy đã được tạo trong không gian tên dưới không gian tên của client. Thêm vào đoạn mã sau trong:

```
using QuotableQuotesClient.QuoteServiceWebReference;
```

Thao tác đơn giản cần để tạo một thể hiện của lớp proxy, gọi phương thức `GetQuote`, và hiển thị thông tin trích dẫn trong giao diện đồ họa. Thực thi thao tác bằng cách sử dụng đoạn mã trong Listing 7.5.

Listing 7.5

```
private void btnGetQuote_Click(object sender, System.EventArgs e)
{
  QuoteService qs = new QuoteService();
  Quote quote = qs.GetQuote();

  if(null == quote)
  {
    MessageBox.Show("An error occurred retrieving a quote");
```

```
    Return;
  }

  UpdateQuoteUI(quote);
}
```

`UpdateQuoteUI` là phương thức trợ giúp trích đoạn dữ liệu từ đối tượng `Quote` và cập nhật ứng dụng giao diện đồ họa. Listing 7.6 chứa đoạn mã cho phương thức `UpdateQuoteUI`.

Listing 7.6

```
private void UpdateQuoteUI(Quote quote)
{
  lblQuote.Text = quote.String;
  lblAuthor.Text = "- " + quote.Author;
  lblDate.Text = ( quote.Date == "Unknown" ) ?
    string.Empty :
    quote.Date;
}
```

7.3.4 Asynchronous Consumption of the Simple Web Service

QuotableQuotes XML Web service đã được sử dụng trong quản lý đồng bộ. Đối tượng proxy được tạo, và phương thức `GetQuote` Web được gọi. Mã hóa sau đó đợi cho phương thức `GetQuote` trả lời. Trong khi công việc nhận được hoàn thành, nó luon luon không hành động. Cài đặt, hình dùng client đang cầu khẩn một XML Web service mà khối xử lý ttheo thứ tự yêu cầu.

Lớp proxy cung cấp hai phương thức để thao tác với trạng thái không đồng bộ Web XML service gọi: `eginWebMethod` và `EndWebMethod`. Trong mỗi trường hợp `WebMethod` cùng với tên của phương thức Web. Ví dụ, proxy `QuoteService` tạo ra phương thức `BeginGetQuote` và `EndGetQuote`.

Phương thức `BeginWebMethod` có hai tham số trong phần thêm vào tham số `WebMethod` đưa đến. Cho đến khi `GetQuote` không chấp nhận bất kỳ tham số nào, `BeginGetQuote` chỉ chấp nhận hai tham số. Trước tiên là tham số của `System.AsyncCallback`. Thể hiện phương thức sẽ được gọi trong WebMethod đã được hoàn thành. Tham số thứ hai là kiểu đối tượng và có thể bất kỳ cái gì mà chúng ta muốn thể hiện trạng thái của WebMethod gọi.

Để tạo một trạng thái client không đồng bộ bằng cách tạo một hàm gốc gọi lại trên client. Phương thức phải là public hay static. Nó phải không có giá trị trả ề và chấp nhận một tham số của kiểu `System.IAsyncResult`. Tham số này thể hiện kết quả của async gọi. Nó cho phép truy cập tới trạng tháo chúng ta thông qua `BeginWebMethod`. Thêm các phương thức sau vào client:

```
public static void GetQuoteCallBack(IAsyncResult ar)
{
  MessageBox.Show("GetQuote completed");
}
```

Bây giờ chúng ta thay đổi sự thực thi của nút thao tác gọi trạng thái không đồng bộ XML Web service. Trước tiên thay thế thực thi hiện tại của nút thao tác cùng với sự thực thi từ mã Listing 7.7:

Listing 7.7

```
private void btnGetQuote_Click(object sender, System.EventArgs e)
{
  QuoteService qs = new QuoteService();
  // Set the url of the proxy to the proper url of the web service

  AsyncCallback getQuoteCB = new AsyncCallback(
    QuotableQuotesClient.Form1.GetQuoteCallBack);

  object[] callBackState = {qs, this};
  qs.BeginGetQuote(getQuoteCB, callBackState);
}
```

Đoạn mã trước tạo một thể hiện của Web service. Đối tượng `AsycnCallback` được tạo thể hiện một con trỏ phương thức `GetQuoteCallBack` trên client. Cuối cùng Web service gọi bắt đầu sử dụng phương thức `BeginGetQuote`. Phương thức này trả về kết quả trước khi phương thức Web gọi hoàn thành.

Listing 7.8 chứa đựng sự thực thi của phương thức `GetQuoteCallBack`.

Listing 7.8

```
public static void GetQuoteCallBack(IAsyncResult ar)
{
   object[] callBackState = (object[])ar.AsyncState;
  QuoteService qs = (QuoteService)callBackState[0];
  Form1 app = (Form1)callBackState[1];

  Quote quote = qs.EndGetQuote(ar);
  if(null == quote)
  {
    MessageBox.Show("No quote object received.");
    return;
  }

  app.UpdateQuoteUI(quote);
}
```

7.4 Sử dụng Web Service có sử dụng DataSet

XML Web service đã trả về cấu trúc dữ liệu khách hàng. .NET Compact Framework cung cấp khả năng truyền sự kiện nhiều dữ liệu phức tạp, giống như là `DataSet`. Mặc dù .NET Compact Framework không hỗ trợ kiểu `DataSet`.

Gửi và nhận một `DataSet` thường xuyên được hoàn thành chính xác giống như là gửi cấu trúc dữ liệu hoặc những phần dữ liệu đơn giản, như là chuỗi. Phương thức Web đơn giản cần chấp nhận hoặc trả về đối tượng `System.Data.DataSet`. Trong khi tìm hiểu DataSet đầu tiên, chúng ta gửi hoàn thành một bảng `Quotes` tới client. Cho phép ứng dụng offline và đợi đến khi hiển thị trích dẫn đến khi người sử dụng bấm vào `GetQuote`.

Có một stored procedure trong CSDL `QuotableQuotes` gọi là `GetQuotes`, thủ tục này trả về tất cả thông tin trích dẫn trong bảng `Quotes`. Trở lại phần thiết kế XML Web service, và kéo thủ tục này vào trong phần thiết kế. Thay đổi tên câu lệnh này thành `cmdGetQuotes`. Chúng ta sẽ

đưa ra phương thức Web `GetQuotes` trên `QuoteService`, phương thức này trả về một `DataSet`. `DataSet` sẽ được điền vào bằng cách sử dụng `SqlDataAdapter` chung với đối tượng SqlCommand `cmdGetQuotes`, mà chúng ta thêm vào đối tượng. Được mã hóa như sau:

```
[WebMethod]
public DataSet GetQuotes()
{
  quoteConnection.Open();

  try
  {
    DataSet quotesDS = new DataSet();
    SqlDataAdapter quotesDa = new SqlDataAdapter(this.cmdGetQuotes);

    quotesDa.Fill(quotesDS);

    return quotesDS;
  }
  finally
  {
    quoteConnection.Close();
  }
}
```

Dịch và thử phương thức Web trong cách giống với phương thức `GetQuote` đã được thử. Lần này kết quả sẽ trả về một trang dài XML tới trình duyệt. XML đơn giản dưa ra XML từ `DataSet.WriteXml`. Bởi vì kiến trúc XML Web service tuần tự hóa một `DataSet` bằng cách gọi phương thức `DataSet.WriteXml`.

Bay giờ chúng ta tạo một ứng dụng client sử dụng `DataSet` từ Web service. Bắt đầu bằng cách tạo một Smart Device Application giống như cách tạo ứng dụng `QuotableQuotesClient`. Tạo ứng dụng có giao diện đồ họa và thêm đoạn mã sau vào lớp `Form1`:

```
private DataSet quotesDataSet;
private int curQuoteRowNdx;
```

Bấm đúp vào nút Get Quotes sẽ đưa đến sự kiện click. Listing 7.9 chứa mã nguồn.

Listing 7.9

```
private void btnQuote_Click(object sender, System.EventArgs e)
{
  if(null == quotesDataSet)
  {
    QuoteService qs = new QuoteService();
    // Set the proxy's url property to the correct url of the server

    quotesDataSet =  qs.GetQuotes();
    curQuoteRowNdx = 0;
  }

  if( quotesDataSet.Tables.Count <= 0 )
  {
    MessageBox.Show("Could not retreive the quotes dataset.");
    return;
  }

  if(curQuoteRowNdx >= quotesDataSet.Tables[0].Rows.Count)
    curQuoteRowNdx = 0;
```

```
  DataRow quote = quotesDataSet.Tables[0].Rows[curQuoteRowNdx];
  curQuoteRowNdx++;

  UpdateQuoteUI(quote, 0);
}
```

Quotes DataSet được download từ dịch vụ web khi quotes DataSet là null. Lần đầu tiên nút Get Quotes được bấm DataSet có giá trị null. Các lần sau được gọi thông tin trích dẫn được lưu trong quotes DataSet cục bộ , vì vậy không cần thiết download lại.

Nếu quotes DataSet có giá trị null, sau đó đối tượng QuoteService proxy được tạo, thuộc tính URL của proxy được cấu hình, và phương thức Web GetQuotes được gọi.

Tương ứng với DataRow được nhận về từ DataSet và UpdateQuoteUI được gọi, cái này hiển thị thông tin trích dẫn tới ngườiu sử dụng. UpdateQuoteUI được thực thi để sử dụng một DataRow thay cho cấu trúc thông tin trích dẫn. Có một tham số giá trị nguyên có tên offset kiểu Int32.

Listing 7.10 chứa đựng sự thực thi của phương thức UpdateQuoteUI.

Listing 7.10

```
private void UpdateQuoteUI(DataRow quote, int offset)
{
  lblQuote.Text = (string)quote[offset];
  lblAuthor.Text = "-" + quote[offset + 1];
  lblDate.Text = "Unknown" == (string)quote[offset + 2] ?
    string.Empty :
    (string)quote[offset + 2];
}
```

Dịch và chạy ứng dụng. Ứng dụng sẽ hành động chính xác như nó đã làm trước, ngoại trừ trích dẫn sẽ lặp lại ngoại trừ hiển thị theo thứ tự không định trước.

7.5 Sử dụng Web Service trả về kiểu DataSet

.NET Compact Framework không hỗ trợ kiểu DataSet. Nếu Web service sử dụng kiểu DataSet, .NET Compact Framework clients sẽ không sử dụng được Web service. Đặc biệt, client sẽ bị lỗi khi cố gắng dịch proxy vào ứng dụng.

Có công việc xung quanh vấn đề sử dụng kiểu DataSet:

- Thay đổi XML Web service để sử dụng DataSet chuẩn.
- Thay đổi XML Web Service để đưa ra DataSet chuẩn.
- Tạo một XML Web service mới sử dụng kiểu DataSet nguyên bản và chuyển nó DataSet chuẩn.

Có hai lựa chọn có thể dường như giống nhau, nhưng chúng khác nhau. Lựa chọn đề xuất là chúng ta loại bỏ toàn bộ tất cả thể hiện của kiểu DataSet từ Web service và sử dụng DataSet chuẩn trong khi mã hóa. Đề xuất thứ hai chúng ta có thể thay đổi Web service để đưa ra

(send/receive) chỉ DataSet chuẩn. Điều này có nghĩa là trong mã nguồn có thể tạo và sử dụng tiện ích kiểu DataSet, nhưng phương thức Web chỉ có thể sử dụng DataSet chuẩn như là tham số giá trị trả về.

Đơn giản là ghi ra một XML Web service để đưa ra DataSet chuẩn nhưng bên trong sử dụng kiểu DataSet. Trở lại đối tượng QuoteableQuotes XML Web service, và phần thiết kế XML Web Service designer. Xác định bảng Quotes trong Server Explorer. Nó sẽ như sau đây: <Server Name>\SQL Servers\<Server Instance Name>\QuotableQuotes\Tables. Kéo bảng Quotes vào phần thiết kế. Nó tạo ra một SqlDataAdapter, để tạo một kiểu DataSet cho bảng Quotes. Đổi tên SqlDataAdapter thành daQuotesDS.

Bấm vào SqlDataAdapter, vào phần menu Data, và chọn mục Generate DataSet . . . Nó sẽ đưa đến hộp thoại Generate DataSet. Thay đổi tên của DataSet mới thành QuotesDataSet, và bấm OK. Nó sẽ sinh ra kiểu DataSet cho bảng Quotes.

Tạo một phương thức Web tên là GetTypeQuotes. Phương thức Web sẽ trả về DataSet chuẩn, nhưng bên trong nó sẽ sử dụng QuotesDataSet kiểu DataSet. Listing 7.11 chứa đựng một phương thức sẽ chuyển kiểu DataSet thành DataSet chuẩn trước khi trả về người sử dụng.

Listing 7.11

```
[WebMethod]
public DataSet GetTypedQuotes()
{
  quoteConnection.Open();

  try
  {
    QuotesDataSet quotesDS = new QuotesDataSet();
    this.daQuotesDS.Fill(quotesDS);

    return quotesDS;
  }
  finally
  {
    quoteConnection.Close();
  }
}
```

Để kết thúc ví dụ về Quotable Quotes DataSet client cần phải thay đổi để sử dụng phương thức GetTypedQuotes. Chỉ thao tác click của nút cần thiết phải thay đổi. Thao tác sẽ gọi GetTypeQuotes thay cho GetQuotes. Mặc dù, tham số offset của phương thức UpdateQuoteUI sẽ thay cho 1 thành 0 theo thứ tự bù cho cột ID trong DataRow. Listing 7.12 chứa đựng đoạn mã mới cho phương thức này.

Listing 7.12

```
private void btnQuote_Click(object sender, System.EventArgs e)
{
  if(null == quotesDataSet)
  {
    QuoteService qs = new QuoteService();
```

```
    // Set the proxy's url property to the correct url of the server

    quotesDataSet =  qs.GetTypedQuotes();
    curQuoteRowNdx = 0;
  }

  if(quotesDataSet.Tables.Count >= 0)
  {
    MessageBox.Show("Could not retreive the quotes dataset.");
    return;
  }

  if(curQuoteRowNdx >= quotesDataSet.Tables[0].Rows.Count)
    curQuoteRowNdx = 0;

  DataRow quote = quotesDataSet.Tables[0].Rows[curQuoteRowNdx];
  curQuoteRowNdx++;

  UpdateQuoteUI(quote, 1);
}
```

7.6 Tổng kết

- .NET Compact Framework cung cấp khả năng cho ứng dụng sử XML Web services.
- .NET Compact Framework cung cấp khả năng cho sử dụng dữ liệu đơn giản, như là chuỗi, dữ liệu phức tạp, như là cấu trúc dữ liệu người dùng tự định nghĩa và DataSet.
- .NET Compact Framework hỗ trợ đồng bộ và không đồng bộ gọi XML Web service.